NETWORK FORMATION AND ROUTING FOR MULTI–HOP WIRELESS AD–HOC NETWORKS

A Dissertation
Presented to
The Academic Faculty

By

Xin Zhang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
August 2006

NETWORK FORMATION AND ROUTING FOR MULTI–HOP WIRELESS AD–HOC NETWORKS

Approved by:

Dr. George F. Riley, Committee Chair
Asst. Professor, School of ECE
Georgia Institute of Technology

Dr. Henry L. Owen
Professor, School of ECE
Georgia Institute of Technology

Dr. Douglas M. Blough
Professor, School of ECE
Georgia Institute of Technology

Dr. Richard M. Fujimoto
Professor, College of Computing
Georgia Institute of Technology

Dr. Raghupathy Sivakumar
Assoc. Professor, School of ECE
Georgia Institute of Technology

Date Approved: May 3, 2006

To my parents,

Renjun Zhang and ZiQiang Zhu.

ACKNOWLEDGMENTS

First of all, I would like to express my special and sincere gratitude to my dissertation advisor, Dr. George Riley, who has been constant in his valuable guidance and encouragement in my PhD research. I feel extremely lucky to be guided by his insight in research issues as well as great patience and care. Without his help, I could not have completed this dissertation successfully.

I would also like to extend my appreciate to Dr. Douglas Blough, Dr. Raghupathy Sivakumar, Dr. Henry Owen, and Dr. Richard Fujimoto for reviewing this dissertation and serving on my PhD dissertation committee. Their comments and suggestions have helped improve the quality of this dissertation.

I am grateful to my colleagues in the MANIACS Laboratory at Georgia Institute of Technology for their friendship, help, and support.

I would like to express my sincere gratitude to my parents, Renjun Zhang and ZiQiang Zhu, and my elder sister, Yun Zhang, whose love and encouragement have been continuously given me the braveness to face the challenges and finish my PhD successfully. Last but not lease, I am deeply thankful to my husband, Liang Cheng, for his love, support as well as precious discussions about my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Wireless ad hoc networking is an important part of the next generation wireless network architecture. Based on the distance over which data is transmitted, wireless networks can be classified into wireless wide area networks (WWANs), wireless metropolitan area networks (WMANs), wireless local area networks (WLANs), and wireless personal area networks (WPANs). Wireless personal area networks (WPANs) were originally designed to allow convenient interconnection of devices around an individual person or computer. Presently, the digital health care and wireless sensor networks are among the major applications.

The first attempt to define a standard for WPANs dates back to Ericsson's Bluetooth project to enable communication between mobile phones using low–power and low–cost radio interfaces. Bluetooth is a promising short–range wireless communication technology with the characteristics of interference resilience and power efficiency both of which are desirable for wireless personal area networks or sensor networks. The new Intel Mote sensor devices have Bluetooth technology incorporated as the standard wireless communications interface.

When using Bluetooth in applications where multi–hop routing is required, groups of Bluetooth piconets (one master and at most seven active slaves form a piconet) combine together to form a scatternet. However, most of the existing scatternet formation protocols are designed to facilitate communications between any two pairs of devices, regardless of the actual traffic demand pattern. For wireless personal area network or sensor network applications with low duty cycle traffic patterns, an on–demand scatternet formation protocol can achieve significant power saving by avoiding unnecessary network connectivity. To that end, this research introduces an on–demand scatternet and route formation protocol designed specifically for Bluetooth–based wireless sensor networks with many–to–one communication architectures.

The objective of this research is to build a scatternet on demand by taking into account network architecture and traffic pattern. The scatternet formation protocol is also able to cope with multiple sources initiating traffic simultaneously as well as prolong network lifetime. A modified *Inquiry* scheme using extended ID packet is introduced for fast device discovery and power efficient propagation of route request messages with low delay. A mechanism employing *POLL* packets in *Page* processes is proposed to transfer scatternet formation and route reply information without extra overhead. In addition, the energy aware forwarding nodes selection scheme is based on local information only and results in more uniform network resource utilization and improved network lifetime. Simulation results show that this protocol can provide scatternet formation with reasonable delay and with good load balance which results in prolonged network lifetime for Bluetooth–based wireless sensor networks.

The Intel Mote is a new sensor node platform with improved radio bandwidth and reliability due to the usage of Bluetooth radio. The connection–oriented nature of Bluetooth raises the issues of effective multi–hop network (scatternet) formation and maintenance that network and routing layer must address on top of the TinyOS abstractions. The hop distance and wireless link quality pose major challenges to multi–hop network performance, especially for connection–oriented networks such as Bluetooth scatternet. In this research, a metric–based scatternet formation algorithm for the Intel Mote is presented. It optimizes the Bluetooth network formation from the hop distance and link quality perspectives. In addition, a smart repair mechanism is proposed to deal with link/node failure and recover the network connectivity promptly with low overhead. The experiments with the Intel Mote platform demonstrate the effectiveness of the optimizations, which make the platform more powerful. The concepts of this work can be applied in a straightforward manner to other Bluetooth devices.

As wireless devices become more and more popular, ad hoc networks grow in the number of nodes as well as the complexity of communication among the large number of nodes.

This research also investigates the performance and scalability of ad hoc routing protocols in very large–scale wireless ad hoc networks. Due to the limitation of simulation technologies, it is either impossible or very hard to investigate the scalability of ad hoc routing protocols in very large–scale wireless networks. However, in this research, a comprehensive simulation study is conducted of the performance of an on–demand routing protocol on a very large–scale, with as many as 50,000 nodes in the network. The scalability analysis is addressed based on various network sizes, node density, traffic load, and mobility. The reasons for packet loss are analyzed and categorized at each network layer. Based on the observations, we observe the effect of the parameter selection and try to exhaust the scalability boundary of the on–demand routing protocol for wireless ad hoc networks.

CHAPTER 1

INTRODUCTION

1.1 Background

Wireless Personal Area Networks (WPANs) address wireless ad hoc networking of portable and mobile devices, allowing these devices to communicate and interoperate with one another [1]. Bluetooth [2] is a standard for short–range wireless communications in the range known as WPAN [3, 4]. Bluetooth devices use the 2.4 GHz band, which is unlicensed in most countries (in the United States it is known as the industrial, scientific, and medical, (ISM) band). The bandwidth is divided into 79 1-MHz-wide channels. The channels are accessed using an Frequency Hopping Spread Spectrum (FHSS) technique, with a signal rate of 1 Mb/s.

1.1.1 Network Architecture

- **Piconet –** Piconet is the basic element in Bluetooth technology. In Bluetooth specification, a master/slave architecture is defined. A maximum of eight Bluetooth devices, which include seven active slaves and one master, working together to form a piconet. A piconet is the simplest configuration of a Bluetooth network.

  If a master has more than seven slaves, some of the slaves have to be put into Bluetooth low power mode park. To communication with parked slaves, the master has to unpark it, while possibly parking another slave. The park action is time and power consuming, and should be avoided whenever possible.

  The piconet formation phase concerns the assignment of roles, either master or slave, to all the nodes in the network. A node can only be master in one piconet, while it can be enrolled as slaves in multiple piconet.

- **Scatternet –** A scatternet is a topology over which multi–hop wireless communication can be built with Bluetooth technology. Multiple piconets are linked together

by using some nodes as bridges. The bridge nodes belong to multiple piconets . A node can be a master in one piconet at most and a slave in several others. A scatternet configuration is shown in Figure 1. Two types of bridges, namely master/slave bridge and slave/slave bridge are shown in Figure 2. By connecting neighboring piconets, Bluetooth devices establish multi–hop networks, which spread out in a large geographic area. In this sense, Bluetooth provides the wireless extension to the various heterogeneous network architectures, which enables a more pervasive wireless access.

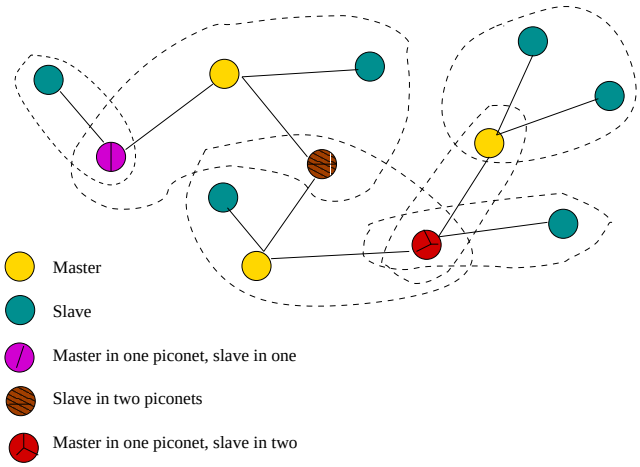


Figure 1. A scatternet configuration

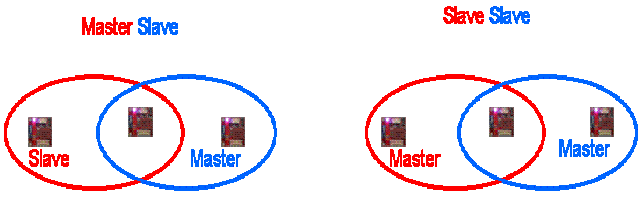


Figure 2. Bridge types

The Bluetooth specification only defines the configuration of piconets and leaves the scatternet formation an open research issue.

1.1.2 Connection Establishment

- **FHSS –** The multiplexing technology in Bluetooth is frequency hopping spread spectrum (FHSS). Bluetooth devices access the different radio channels according to an extremely long pseudo–random sequence that is generated from the address and clock of the master in the piconet. Different piconets use different hop sequences, which leads to interference resilience.

  The frequency hopping rate is 1600 hops per second, which means that the transmission frequency remains unchanged for 625μs slot. The master transmits in the even slots, while the slaves transmit in the odd ones.

- **Inquiry and Page –** The first step for Bluetooth devices to be grouped into piconets and form scatternet thereafter is to be aware of their neighbors. However, unlike other broadcast wireless technologies using one channel, such as 802.11, Bluetooth's master/slave architecture requires both time synchronization and frequency synchronization during its device discovery. Two devices are time synchronized when they are in opposite states (*Inquiry* for master and *Inquiry Scan* for slave) and synchronized in the transmission/reception schedule. The frequency synchronization means two devices hop to the same frequency at the same time. When both of these synchronizations occur, a communication link can be established. The synchronization requirements imply that Bluetooth devices that are in each other's radio range may not know about the others' existence if they are unsynchronized. In this case, physical proximity of Bluetooth devices does not mean the existence of communication links.

  Because Bluetooth is based on connection–oriented communication, two Bluetooth devices need to go through *Inquiry* and *Page* phases before they can communicate with each other. The *Inquiry* phase let the master get the device address and clock

value of the slave. After that, in the *Page* phase, the two Bluetooth devices synchronize over the frequency hopping sequence, which is unique and decided by the master. Once the connection is established between the devices, they can communicate in a time division duplex mode.

1.2 Motivation and Solutions

Advances in micro–sensing technology as well as numerous novel applications have led to a substantial volume of research on wireless personal area networks and wireless sensor networks (WSNs) [5, 6, 7]. One example of the important applications for WSNs is health care. WSNs can deliver quality care to a rapidly growing population of older adults and benefit the patients by using sensors that monitor vital signs. For example, Intel has developed systems to follow activities of people with Alzheimer's disease. These systems are built by adopting the mote technology which has Bluetooth equipped as the wireless communication interface [8]. Other health care applications such as IBM m-Health wireless health solutions also rely on Bluetooth to transfer sensor data [9].

Other protocols, such as IEEE 802.11 [10], use idle listening and collision avoidance actions, making such protocols power consuming and less appropriate for WSNs. Newer protocols, such as IEEE 802.15.4 [1], may be appropriate for WSN applications, but as yet devices with such interfaces are not widely available. Interconnecting these sensor devices with external networks is difficult. And some WSNs applications require transmission of high data rate audio or video data, which is beyond the range of low data rate protocol such as IEEE 802.15.4. In this case, Bluetooth may be a good choice in balancing the data rate and power consumption.

In [11, 12], various advantages and limitations are discussed concerning Bluetooth–based WSNs. The main advantages of employing Bluetooth are as follows:

- **Ubiquity –** Most sensor network nodes, e.g. the motes, have limited resources for

data storage, processing, and power. They rely on base stations or gateways to connect to external networks for tasks requiring more storage or higher performance. Such gateways with Bluetooth interfaces are widely spread. If sensor nodes are equipped with Bluetooth, they can communicate seamlessly with these ubiquitous gateways. In addition, some sensor network applications need immediate reactions taken by actuators when certain events are sensed by sensors. If Bluetooth is supported by the sensor nodes, many commercial products with Bluetooth interfaces can be used as such actuators. Furthermore, using Bluetooth as the radio for sensor nodes enables easy communication with personal devices such as PDAs and laptops since many of them have Bluetooth interfaces. This allows direct user interaction with sensor nodes for data collection, controlling, and debugging.

- **Power Efficiency –** For wireless sensor networks, low power consumption is essential. The Bluetooth protocol allows the radio to enter low power modes while maintaining synchronization when no transmission or reception is active on a communication link. These modes greatly reduce the power consumption. The low power modes used by Bluetooth, listed in increasing order of power efficiency, are sniff mode, hold mode, and park mode.

- **Interference Resilience –** Wireless sensor networks consist of a large number of elements. Radio–level interference can be a significant problem when sensor nodes within communication range compete for a shared channel. Bluetooth, on the other hand, exploits frequency hopping spread spectrum. Sensor nodes within communication range can use separate channels to transmit data. Bluetooth's resilience to interference makes it a good candidate in the context of sensor networks.

- **High Data Rate –** There are two types of links defined in Bluetooth, asynchronous connectionless link (ACL) and synchronous connection–oriented link (SCO). SCO links provide constant bit rate symmetric channels, which have guaranteed delay

and bandwidth. It is suitable for voice applications that require fixed bandwidth. ACL links are appropriate for data traffic. The asymmetric data rate of ACL links can be up to 723kbps. Comparing to IEEE 802.15.4 which has a typical data rate of 250kbps, Bluetooth's high data rate makes it suitable for many wireless sensor network applications.

- **Security –** Bluetooth provides security at both application layer and link layer. Four different entities are used for maintaining security at the link layer, the Bluetooth address which is unique for each device, two secret keys, and a random number which is different for each new transaction. These embedded security mechanisms make Bluetooth a good candidate for WSNs within which security is an essential requirement.

Given these desirable properties for wireless sensor network applications, several sensor devices using Bluetooth as the wireless interface have been introduced, including BTnodes [14] from ETH Zurich and the Intel Mote [15, 16, 17].

The applications for WSNs are widespread and have diverse requirements. The protocols for WSNs are more application–driven rather than universal. An important class of WSN applications have a multi–level network architecture with a large number of sensor nodes dispersed within a geographic area that often communicate to the external network through a sink node. Since the transmission range of Bluetooth devices is only tens of meters, the communication between the sensor nodes and the sink node will often require multi–hop routing. WSN applications with this structure include habitat monitoring [18, 19], civil infrastructure health monitoring [20], data collection [21], weather monitoring and reporting, and others. In these applications, data transfers occur infrequently, with a few unpredictable bursts. This type of traffic pattern is ideal for Bluetooth–based WSNs [11, 12]. Bluetooth connections for such applications are established as needed, depending on the traffic requirements, and are torn down when traffic ceases in order to save power.

One of the primary requirements for Bluetooth–based WSNs to work effectively is an efficient scatternet formation and routing protocol for multi–hop networks. Connectivity is the principle concern in most of the existing scatternet formation algorithms. However, in WSNs with a many–to–one (sensors to sink) communication architecture and the burst traffic pattern, a more efficient scatternet formation protocol is in demand. Such protocols become even more complicated when multiple sources start the scatternet formation and route discovery concurrently, which involves common intermediate nodes. In addition, in wireless sensor networks where sensor motes are powered by batteries, uniform battery drain on all nodes is desirable and will lead to longer network lifetimes.

In this dissertation, an on–demand scatternet formation and routing protocol designed specifically to address these requirements is proposed. A modified *Inquiry* method with extended ID (EID) packets is used in route discovery for asymmetric device discovery. During the scatternet formation and route reply phase, modified *POLL* packets are used in the *Page* mode. Further, the resulting scatternet maintains cross routes for multiple sources initiating traffic at the same time. To balance network load, a local decision of whether to forward a received route request is made by each node based on its own residual energy. The energy-aware intermediate node selection prolongs the network lifetime without the overhead of neighbor information exchange.

Intel Mote is a new platform design that delivers a high level of integration as well as low–power operation in a small physical size [13]. They can serve as the building blocks for wireless sensor networks. These networks are being deployed in a growing number of applications. Intel Mote has Bluetooth equipped for wireless communication.

The existing scatternet formation algorithm in Intel Mote is a simple scheme without taking into consideration of link quality, hop distance, and reliability. Therefore, the performance of such scatternet in the environment with lossy wireless links is poor. In this dissertation, a metric–based scatternet formation scheme with cross layer design is proposed. In addition, an efficient network recovery algorithm is proposed for network maintenance.

With these optimizations incorporated in the Intel Mote platform, the performance of the constructed scatternet is improved significantly.

Besides the MAC layer, efficient routing protocols are equally important for wireless networks. As the network size becomes larger and larger, the scalability comes out as an essential factor. In this dissertation, the potential scalability of an on–demand ad hoc network routing protocol, specifically the Ad hoc On–Demand Distance Vector (AODV) routing protocol is evaluated. The reasons for packet loss are analyzed and categorized in detail at each network layer. The purpose of this work is trying to exhaust the scalability limits of the on–demand routing protocols in various scenarios for wireless ad hoc networks. It provides a guideline for the protocol design and future enhancements.

1.3 Dissertation Outline

The remainder of the dissertation is organized as follows. In Chapter 2, an energy–aware on–demand scatternet formation and routing protocol for the many–to–one network architecture is proposed, which includes the description of the Bluetooth simulator in *GTNetS*, the scatternet formation protocol, as well as the performance evaluation of the scatternet formation and routing protocol. In Chapter 3, the metric–based scatternet formation and recovery optimization for Bluetooth–based sensor devices is proposed. The performance comparison to the schemes without the optimization is described based on the experiments with an implementation in Intel Mote platform. Chapter 4 discusses the scalability of ad hoc on–demand routing protocol in very large–scale mobile wireless networks. Finally, Chapter 5 concludes the dissertation and summarizes the results.

CHAPTER 2

ENERGY–AWARE ON–DEMAND SCATTERNET FORMATION AND ROUTING PROTOCOL FOR MANY–TO–ONE AD HOC NETWORKS

2.1 Introduction

2.1.1 Problem Description

Bluetooth was initially designed as a cable interconnect replacement technology. Thus, connectivity is the primary concern for most of the existing work on multi–hop network construction (scatternet formation) schemes. However, for the low–power wireless sensor networks with low duty cycle traffic, maintaining the connection of the entire network is a significant power drain. In addition, unlike Bluetooth ad hoc networks with random communication pattern among the nodes in the networks, many–to–one communication architecture is typical in wireless sensor networks. Hence, on–demand scatternet formation taking into account the structure and traffic patterns of wireless sensor networks leads to a more energy–efficient network.

In the Bluetooth device discovery phase (*Inquiry*), a master is able to get the device addresses and clock values from the slaves, whereas slaves have no information about the master. This is the so–called asymmetric device discovery. Due to this asymmetric property, temporary piconets are formed in multi–hop Bluetooth networks, and L2CAP messages are exchanged in order to achieve mutual knowledge. These temporary piconets are required in both route request phase to transfer source and last hop information, as well as route response phase to transfer master/slave role assignment information. The temporary piconet formation and tearing down result in lengthy scatternet formation time and excessive message overhead.

The problem of efficient scatternet formation also involves multiple nodes initiating the scatternet formation simultaneously with common intermediate nodes on the cross routes. The traditional solution to this problem is that one node devotes to a single branch of the

scatternet formation, and cross routes are handled serially. In densely deployed wireless sensor networks with multiple simultaneous data sources, this sequential scheme results in long scatternet formation delay. A more efficient solution to the Bluetooth scatternet formation involving common intermediate nodes is required.

2.1.2 Related Work

In the literature of Bluetooth scatternet formation, the major solutions can be categorized as proactive and on–demand mechanisms.

The efforts to form a multi–hop Bluetooth network proactively include Bluetree [22], which initiates the scatternet formation by a designated root node and generates a tree-like scatternet topology. In Bluenet [23], some nodes become masters randomly and form piconets; then, neighboring piconets form a scatternet. All piconets in Bluenet have a bounded number of slaves. Bluestars [24] selects master nodes based on the knowledge of neighbors and may have more than seven slaves within a piconet. So it requires putting some nodes in low power mode park and the protocol complexity is increased. Bluemesh [25] forms scatternet topologies with multiple paths between any pair of nodes. All of the above schemes try to form the scatternet including every node within a Bluetooth network. The performance comparison of some of these protocols is presented in [26].

To the best of our knowledge, the only existing work addressing Bluetooth on–demand scatternet formation is presented in [27, 28, 29].

In [27], an extended ID (EID) connectionless broadcast mechanism is introduced. The route discovery delay is greatly reduced compared with traditional Bluetooth broadcast in the L2CAP layer. However, the ID packet in Bluetooth is designed to be small initially to save power since the number of ID packets transmitted in the Bluetooth *Inquiry* phase is very large (two ID packets per 625 μs time slot). Substituting all ID packets with much longer EID packets to transfer source information in scatternet formation is unnecessary and power consuming because most of the ID packets are just for neighbor probing and synchronization. In addition, the simulations of route discovery in [27] only consider the

scenario of a single source. When multiple sources in the network initiate the scatternet formation and route discovery simultaneously involving common intermediate nodes, they will interfere with each other and degrade the performance significantly. In this case, the concurrent instead of consecutive cross route formation needs to be addressed.

Another on–demand Bluetooth scatternet formation algorithm (ODBT) is presented in [28]. ODBT constructs a scatternet with a tree topology. It is an extension to Bluetree [22], with the ability to cope with Bluetooth devices dynamically joining and leaving the scatternet. However, it still tries to connect all the nodes within the network and cannot operate in the presence of multiple sources simultaneously starting the formation of a scatternet involving the same Bluetooth devices.

In [29], a two–phase scatternet formation (TPSF) protocol is introduced to support dynamic topology changes. A control scatternet is constructed in the first phase to support topology changes and route determination while an on–demand scatternet is created in the second phase whenever data communication is needed. Maintaining the control scatternet constructed in the first phase is power consuming and makes TPSF similar to proactive approaches.

2.1.3 Solution Outline

The on–demand scatternet formation protocol proposed in this dissertation shares some common points with [27]. The Baseband layer broadcast, instead of the L2CAP layer broadcast, is used for flooding the scatternet and route formation requests to achieve reduced delay. However, a modified *Inquiry* with EID packets is introduced in the route discovery phase for power saving. Moreover, a modified *POLL* packet in *Page* is employed during the route reply phase. The proposed protocol takes into the consideration of the special many–to–one architecture of wireless sensor networks as well as power conservation objective. Other than the on–demand scatternet formation work mentioned above, this proposed protocol provides the mechanism to deal with multiple sources initiating the formation of a scatternet simultaneously, which is essential for dense sensor networks and

makes in–network data processing simple. The intermediate Bluetooth nodes residing on cross routes switch between *Inquiry Scan* and *Page Scan* states. Therefore, the cross routes can be established concurrently. Furthermore, maximizing the network lifetime by avoiding energy depletion at some critical nodes is considered in the proposed scatternet formation and routing mechanism. A detailed description of the protocol as well as the Bluetooth simulator in the Georgia Tech Network Simulator (*GTNetS*) are presented in the following sections.

2.2 Bluetooth Simulator in GTNetS

In order to evaluate the performance of Bluetooth scatternet protocol, we first designed a detailed Bluetooth simulator [30] in the the *Georgia Tech Network Simulator* (*GTNetS*) [31, 32, 33] developed by our research group. *GTNetS* was designed for efficient simulation of large–scale networks. It achieves good scalability by using distributed simulation methods, as well as an efficient design for both memory and computational resources [32]. The protocol stack architecture in *GTNetS* maps exactly to real networks and hardware. Each protocol stack is implemented as a stand–alone class in object–oriented C++. It is easy to understand and extend based on the existing simulation models. *GTNetS* supports a number of protocols at different layers.

Our Bluetooth simulator takes advantage of the scalability of *GTNetS* as well as layered design architecture, and extends it for sensor network and Bluetooth network simulations. One can easily make further modifications to the Bluetooth protocols based on our Bluetooth simulator, in order to validate the performance and suitability of TDMA FHSS based schemes for sensor networks. Details on our implementation of the Bluetooth protocol in the *GTNetS* simulator are given in the following sections.

2.2.1 General

The Bluetooth stack and *GTNetS* Bluetooth modules are shown in Figure 3. In *GTNetS*, a *Node* object represents the functionality of a network node. We derive the Bluetooth

BlueNode class from *Node* class. Besides *Node's* common functionality, a *BlueNode* has a 48-bit Bluetooth device address, which identifies a Bluetooth device uniquely. Another extension of *BlueNode* is that it has a neighbor list. This neighbor list is used to store all the *BlueNodes* found within its radio range during the *Inquiry* phase. The neighbor list is checked and updated each time before a transmission occurs to insure reachability. In addition to the address information of the neighbor *BlueNodes*, the estimated clocks of these neighbors obtained at *Inquiry* are also stored in this list for the purpose of *Page* later. The *BlueNode* derived from *Node* also enables it to inherit the animation feature of *Node* in *GTNetS*.

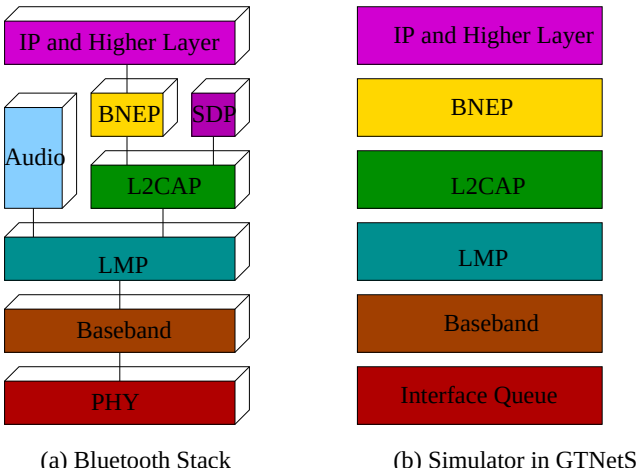


Figure 3. Bluetooth stack and *GTNetS* Bluetooth modules

In *GTNetS*, a packet consists of a stack of *Protocol Data Units* (*PDUs*). Thus it is easy to extend the existing *GTNetS* packets for the Bluetooth simulation by defining all packet headers of the Bluetooth layer modules to be derived from class *PDU*. In this case, when a new packet is generated by the application and moves down to the Bluetooth protocol stack, the header for each layer is generated and pushed into the *PDU* stack. At the destination, the *PDUs* are pop out for processing. The *PDU* stack is shown in Figure 4. An *Freq* field is added to Baseband *PDU* for simulation use. This field has no correspondence to real network packets. The *Freq* field identifies the packet transmission hopping frequency. It is a number between 0 and 78 (we use the 79-hop system as default) corresponding to

the hopping frequency. The fields and length of the packet header at each layer are implemented exactly as the definition in the specification. Therefore, it is easy to understand and simulate the Bluetooth behavior with high accuracy.

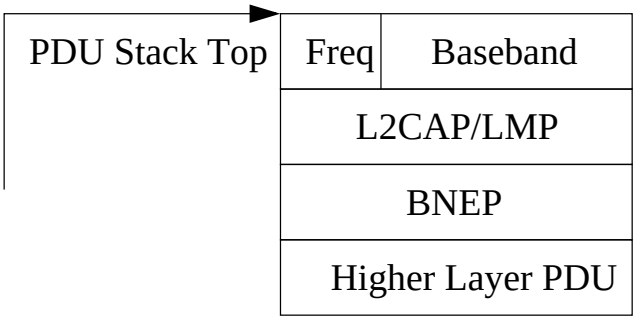


Figure 4. *PDU* stack in *GTNetS* Bluetooth modules

In the following sections, we will describe the Bluetooth protocol implementation in the order from lower to higher layers.

2.2.2 Baseband

The function of Bluetooth Baseband includes:

(i) Discover neighbor nodes by *Inquiry* in order to get the neighbor information including device address and clock.

(ii) Establish frequency hopping communication channel through *Page*.

2.2.2.1 Inquiry and Page

A Bluetooth node has a 48-bit randomly generated device address. The native clock (*ClkN*) at every node is also randomly generated and asynchronous at the beginning. The clock is implemented as a 28-bit counter with the least significant bit (LSB) representing the clock tick of 312.5μs (1/2 of slot time 625μs). The clock counter is updated by scheduling a timer at every clock tick.

During the *Inquiry* process, the master sends out two inquiry trains denoted *A* and *B*. Each train contains 16 distinct hop frequencies. The slaves in the *Inquiry Scan* state listen to one of the 32 hop frequencies the master is sending and change the listening hop frequency

once every 1.28s. To avoid contention when multiple slaves send responses at the same time, a backoff interval between 0 and 1023 time slots is randomly generated by every slave when it gets the *Inquiry* message for the first time. After the timeout of the backoff timer, the slave scans the *Inquiry* message again and transmits response that includes its device address and *ClkN* once the *Inquiry* message is received.

The time for the master to get an *Inquiry* response depends on the clock alignment between the master and the slave as well as the random backoff. From our simulation results, we found 3 to 5 seconds is sufficient for neighbor discovery.

After the *Inquiry* is complete, the master has obtained the device addresses of each of the slaves within its radio range and their native clock values (*ClkN*s). The master can then selectively connect to the slaves by using a *Page* message. The *Page* processing is similar to that of the *Inquiry*, except that the device address is used instead of general inquiry access code (GIAC). In this case, only the slave with the specified device address will respond. The average and maximum time for *Page* is 1.28s and 2.56s respectively assuming the repetition time of each hopping sequence train is 128.

In the Bluetooth simulator for *GTNetS*, we implement all functionality of *Inquiry* and *Page* processes. Four timeout events, *INQUIRY_TO*, *PAGE_TO*, *PAGE_RSP_TO*, and *NEW_CONNECTION_TO*, are scheduled at the appropriate time. After the master gets enough *Inquiry* responses or *INQUIRY_TO* occurs, selective *Page* can follow to establish frequency hopping communication channels.

2.2.2.2 Frequency Hopping

As we mentioned in section 1.2, one of Bluetooth's compelling features for use in sensor networks is its resilience to interference. This comes from the use of frequency hopping spread spectrum (FHSS) scheme. Bluetooth operates at the 2.4GHz ISM band, which is divided into 79 channels with 1MHz spacing. The hopping channel is represented by a pseudo-random hopping sequence hopping through the 79 channels. Each hopping channel is divided into slots with a dwelling time of 625 μs. Since different piconets have different

hopping sequences, there can be multiple piconets existing simultaneously within the same radio region.

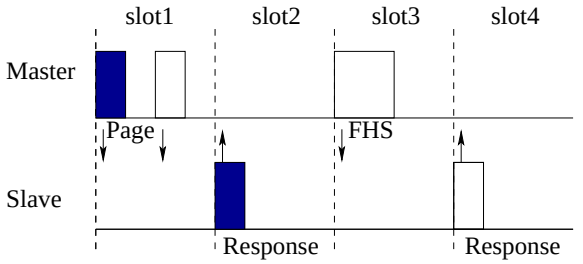
We have implemented a frequency hopping kernel which generates pseudo–random hopping sequences. Our implementation has been verified by comparing to the sample data given in [2]. Three different combinations of addresses and initial clock values were used as the input for this validation test.

As we mentioned earlier, the *Freq* field is inserted into the packet to indicate the transmission frequency. If the packet is composed at its sending slot, the current clock value is used as the input to the frequency hopping kernel. However, in some cases, especially during the connection establishing phase, the control packets are generated as the responses to the received packets and scheduled for transmission in the next time slot in the simulator. Therefore, the corresponding clock counter is adjusted for the correct *Freq* insertion.

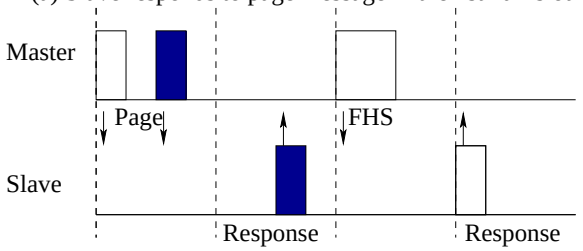
Another synchronization key is to distinguish the responses to *Page* messages sent in the first or the second half of a time slot. Figure 5 illustrates the message sequence for *Page*. During the *Page* process, two ID packets are sent by the master within one time slot. The slave responds either to the first or the second ID packet depending on the hopping frequency matching. The FHS packet from the master always begins from the start of the following slot (slot3 in Figure 5). In this case, the master needs to check the clock value when the response is received in slot 2 of Figure 5 in order to get the correct *Freq* for the FHS packet.

The Bluetooth wireless channel in *GTNetS* is modeled by sending packets to every neighbor node within the transmitter's radio range. When a Bluetooth non–broadcast packet arrives, the receiving node computes its current receiving frequency and compares it to the *Freq* field in the packet header to decide whether this packet should be received and processed.

The Bluetooth simulator in *GTNetS* provides a user friendly animation for frequency hopping. The color of a node is changing with its sending or receiving frequency. So it is



(a) Slave responds to page message in the 1st half slot



(b) Slave responds to page message in the 2nd half slot

Figure 5. Page message sequence

intuitive from the animation to get synchronization information among the nodes within a topology. The screenshot is shown in Figure 6.

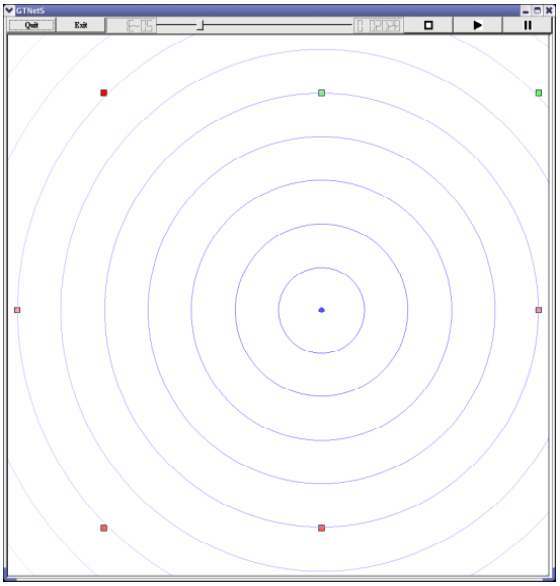


Figure 6. Screenshot

2.2.2.3 *Power Saving Modes*

Power saving is essential for sensor networks since the sensing devices typically use battery power. The Bluetooth design defines methods that can significantly reduce power consumption by allowing the radio to enter low power modes between active communication slots. Since a standard assumption of the duty cycle for a number of sensor network applications is about 1%, the Bluetooth low power modes can be highly efficient. We implemented the three low power modes in Bluetooth, sniff mode, hold mode, and park mode. The implementation of the low power modes is by event scheduling at Link Manager Protocol (LMP) layer and transmission control by checking status words at Baseband layer.

The Bluetooth Baseband power saving modes can reduce power consumption. However, there is a tradeoff between power saving and latency including wakeup latency and buffer latency. Choosing the proper parameters is important to correctly predict power consumption. According to [2], the average clock drift should be less than 20 parts per million (ppm) relative to the slot time of $625\mu s$, and the instantaneous deviation should be less than $1\mu s$ when a node is in active mode. The uncertainty window allowed for master/slave misalignment is $10\mu s$. Thus, the interval (T_{max}) for the master to provide synchronization messages must fulfill (1):

$$\begin{aligned} T_{max} &= \frac{(10\mu s - 1\mu s) * 10^6}{20ppm * 2} \\ &= 225ms. \end{aligned} \tag{1}$$

This indicates that the master should transmit a synchronization message at least every 225ms. When a Bluetooth device operates at low power modes, the clock is driven by a low power oscillator with the accuracy of drift no more than 250ppm and jitter of 10 μs. If the synchronization time for a device returning from low power modes is one time slot ($625\mu s$), the maximum sleep time for a node (S_{max}) is shown in (2):

$$\begin{aligned} S_{max} &= \frac{(625\mu s - 10\mu s) * 10^6}{250ppm + 20ppm} \\ &\approx 2.3s. \end{aligned} \tag{2}$$

Therefore, slaves can sleep within an interval of 2.3 seconds or less. Another consideration for the sleep time is the delay requirements of the applications. The application data from the master to the sleeping slaves must be buffered until the slaves wake up from low power modes. In this sense, the sleep time is application dependent and can be negotiated through Link Manager Protocol (LMP) commands.

In addition to low power modes, we also implemented the Bluetooth power saving in packet level: access code checking. If the access code checking (*CheckAccessCode*) fails, the packet is discarded and the receiver consumes less power than receiving the whole packet.

The Bluetooth simulator in *GTNetS* has been designed from the beginning to model and track power consumption. This measurement is based on power consumption levels: transmitting/receiving, receiving with access code error, active without transmitting/receiving, and low power modes. The power consumption values are from the data sheet of the Bluetooth radio transceiver [42]. Our design is such that it is easy to adjust and extend these parameters when new hardware is introduced.

To investigate the power saving of Bluetooth low power mode, we ran experiments with various senarios. We modeled the traffic generated by the sensor nodes with a simple exponential on–off model. This is a typical traffic type for sensor applications such as habitat and environment monitoring. The on period follows an exponential distribution with its mean value equal to the duty cycle multiplied by the period, while the off period is an exponential distribution with the mean value of $(1 - dutycycle) * period$. To model traffic generated by medium data rate sensors, the traffic generating rate for the on period is set to be $64kb/s$. This is corresponding to the maximum DM1 packet rate ($108.8kb/s$) for Bluetooth Baseband considering BNEP and L2CAP packet headers. Moreover, this is also the worst case scenario for light sensor traffic and the case for aggregated sensor traffic close to the sink. The packet size follows DM1 packet requirement.

Hold Mode For the first set of experiments, we investigate the effect of Bluetooth hold mode on the tradeoff between average packet delay and power saving from entering this low power mode. The active periods between hold modes have the same exponential distribution as the traffic on period. This setting allows the Bluetooth links to deliver the generated traffic in a timely manner. We varied the time during which the nodes stay in hold mode in order to introduce different packet delay. The traffic loads are set to be 1%, 2%, and 5% duty cycle respectively considering the typical assumption for sensor network applications.

Figure 7 shows the percentage of hold time vs. average packet delay. The percentage of time that a node stays in hold mode is in direct proportion to the amount of energy saving. As we list in Table 1, the energy consumption is 20mA when a node is in active mode without any transmission or receiving (ActiveNoTxRx), while it is only $60\mu A$ in low power modes such as hold mode. Putting an idle (without Tx/Rx) node into low power modes wisely is energy efficient since idle in active mode is a major energy drain for typical MAC protocols. The tradeoff for energy saving in the low power mode is sleep delay. In Figure 7, when the traffic load is very light (1% duty cycle), the sensor nodes can enter hold mode for 90% of the entire simulation time while only incur 0.15s average sleep delay per packet. As the traffic load increases, the average sleep delay per packet goes up for a specific percentage of hold time. This is the trend demonstrated by the three curves in Figure 7 representing 1%, 2%, and 5% duty cycle of traffic. However, even with 5% duty cycle of traffic load, the sensor nodes can stay in hold mode for 91% of the simulation time and the average sleep delay is only 1.27s/packet. This level of average packet delay is acceptable for lots of sensor application scenarios. Therefore, putting sensor nodes in hold mode under light traffic can save much energy without introducing large packet delay. In addition, adjusting the time instant to enter hold mode and the duration to stay in hold mode according to the specific application and quality of service requirements can further improve the performance in terms of power consumption and packet delay.

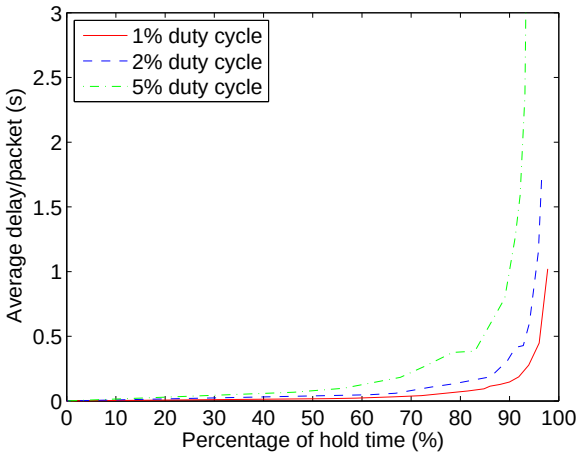


Figure 7. Percentage of hold time vs. average packet delay

Sniff Mode Another set of experiments were conducted to measure the value of energy consumption with and without Bluetooth sniff mode as well as investigate energy distribution on each radio state. These states include transmission (Tx), receiving (Rx), receiving access code error (RxAcErr), active without Tx/Rx (ActiveNoTxRx), and low power modes (sniff, hold, and park). Table 1 shows the average electrical current under different states.

Table 1. Average energy consumption of Bluetooth in various states

State	Current
Tx/Rx	50mA
RxAcErr	24.2mA
ActiveNoTxRx	20mA
Low Power	60μA

In these experiments, we varied the traffic loads from 1% to 20% duty cycle for both scenarios with and without Bluetooth sniff modes and measured the average energy consumption per packet. The parameters for sniff mode are: sniff interval (T_{Sniff}) 1 second, consecutive receiving slot ($N_{SniffAttemp}$) 16 slots, and sniff timeout value ($N_{SniffTimeout}$) 2 slots. The simulation results are shown in Figure 8. In both cases, the average energy consumption per packet decreases as traffic load increases. When the traffic load is light,

without sniff mode, the energy consumption per packet is rather high. This is due to the large amount of energy spent in the state of ActiveNoTxRx. In contrast, under 1% duty cycle traffic load, when the Bluetooth sniff mode with 1 second interval is introduced, the energy consumption per packet in terms of mA is reduced to 51% of the former case without sniff mode. This energy saving benefit of sniff mode can be illustrated by the energy distribution on different radio states shown in Figure 9. In the case of 1% duty cycle of traffic, most of the energy is consumed in the ActiveNoTxRx state rather than spent for Tx/Rx. The sniff mode with the set of parameters declared earlier in this paragraph greatly reduces the energy drain in ActiveNoTxRx state. Therefore, the average energy consumption per packet with sniff mode is significantly reduced.

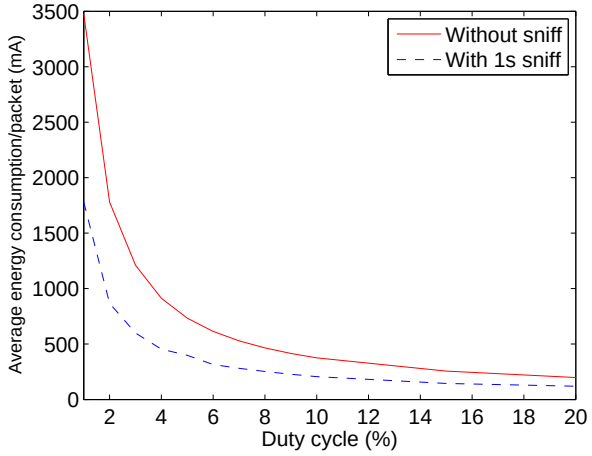


Figure 8. Energy consumption comparison for scenarios with and without sniff

2.2.2.4 Link Control

To provide reliability for the point to point Baseband link, we implemented a link control mechanism. It includes a one bit acknowledgement indication *ARQN* for Automatic Repeat Request (ARQ) and a one bit sequence number *SEQN* in the Baseband packet header.

2.2.2.5 Piconet Management

In a Bluetooth piconet, the master can have up to 7 active slaves. The master communicates with each active slave according to the scheduling scheme. In our simulator, a round–robin

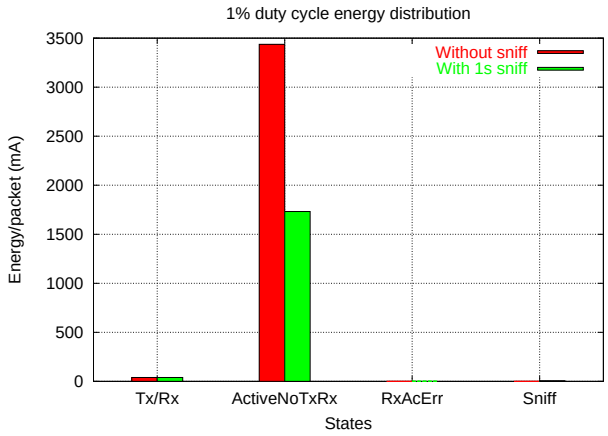


Figure 9. Energy distribution

scheduling scheme is implemented. It is easy to make extension to support other scheduling schemes. To avoid the initiation of multiple link control and link management objects for each master–slave link within a piconet, the concept of context is introduced. The content of the context includes all the information specific to a master–slave link such as active member address, slave device address, *ARQN*, *SEQN*, etc. When the master changes the active slave to which it wants to communicate, the method *SwitchContext* is called. In *SwitchContext*, the current master–slave link parameters are saved and the next scheduled master–slave link parameters are pop out as the current context.

2.2.3 Link Manager Protocol (LMP)

The LMP protocol is used for link initialization, security, and control. The functionality implemented in our Bluetooth simulator includes connection setup/tearing down, link supervision for detecting devices moving out of range, quality of service parameters negotiation, authentication, multislot packets handling, and low power modes switching. LMP packets are differentiated from Logical Link Control and Adaptation Protocol (L2CAP) packets, which may include application data, by one bit in the header and have the higher priority. Thus a separate queue is used for LMP packets.

2.2.4 Logical Link Control and Adaptation Protocol (L2CAP)

The L2CAP layer is above the Baseband protocol and interfaces with upper level protocols as shown in Figure 3. The L2CAP module in our simulator provides protocol multiplexing, segmentation, reassembly, and quality of service configuration.

L2CAP is based on the communication channel model. There are three types of L2CAP channels, as follows. The bidirectional signalling channel, (class *L2capSignalChannel*), is the unique channel between any two entities. It carries all channel control commands. Connection–oriented channels (class *L2capConnChannel*) are used for bidirectional point–to–point connections. Unidirectional connectionless channels provide point–to–multipoint communication. Every endpoint of an L2CAP channel is identified by a logical channel identifier (CID). The signalling channel and connectionless reception channel CIDs are fixed, while the connection–oriented channel CIDs are dynamically allocated by calling the method *AllocateChannelID*. The configuration and data packets exchange are all based on the concept of channels. When an *L2CAP* object is constructed, an *L2capSignalChannel* object is constructed at the same time. The *L2capConnChannel* object is constructed by the corresponding *L2capSignalChannel* when data transmission is required.

2.2.5 Bluetooth Network Encapsulation Protocol (BNEP)

Sensor network nodes employ Bluetooth as the radio media to construct the mesh sensor network and exchange information within it. It is necessary to further extend this network to connect to some external networks, e.g. Internet, through gateways. Bluetooth Network Encapsulation Protocol (BNEP) provides an interface to transport common networking protocols over the Bluetooth media. It provides capacities that are similar to capacities provided by Ethernet. BNEP supports networking protocols such as IPv4 and IPv6.

Our Bluetooth simulator contains a detailed model of the function of BNEP, including BNEP connection control and the interface with L2CAP. With BNEP, it is possible to build a Bluetooth network access point as a bridge between Bluetooth devices and an Ethernet network (for example), as well as sending IP packets between Bluetooth devices.

2.3 Energy–Aware On–Demand Scatternet and Route Formation Protocol

As mentioned in Section 1.2, no protocol for WSNs is universal, but rather the choice of an appropriate protocol is application dependent. In this research, a typical WSN architecture is considered, as shown in Figure 10 [18]. It is popular in habitat and environment monitoring, data collection, etc. Two classes of Bluetooth nodes exist in the network, high power sink node and low power nodes such as Intel motes [15]. Sensor motes communicate with the sink to send data to an external network such as the Internet. Since the sink may not be in the radio range of all the motes, a Bluetooth scatternet must be formed. Considering the typical many–to–one network architecture in WSNs and the uncontinuous traffic demand from sensors, an on–demand scatternet formation protocol would be more energy efficient and would result in improved network lifetime.

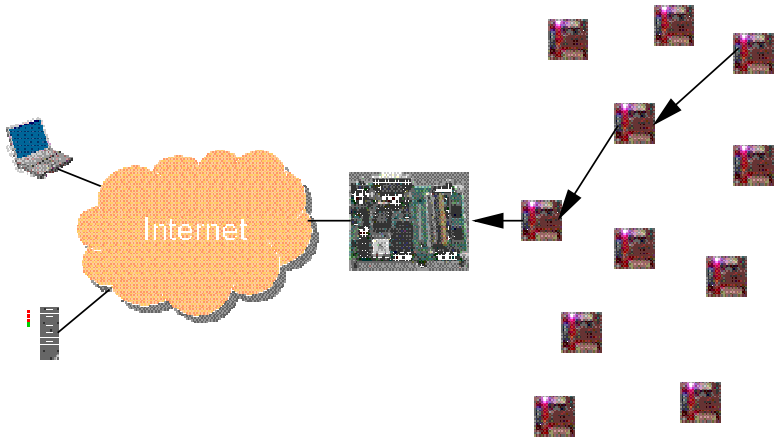


Figure 10. Typical WSN architecture

2.3.1 Protocol Overview

The Bluetooth specification [2] defines a network with a master/slave structure. To interconnect Bluetooth devices into a scatternet, some devices act as bridges between adjacent piconets. In the proposed scatternet formation protocol, slave/slave bridges [1] are chosen to

[1]The node act as slave in both piconets it participates. For example, node 4 in Figure 11 act as slave in piconets of node 1 and node 5.

reduce the number of piconets within a scatternet. Hence, a structure of strict alternating of master/slave roles is maintained along a route from a source to the sink.

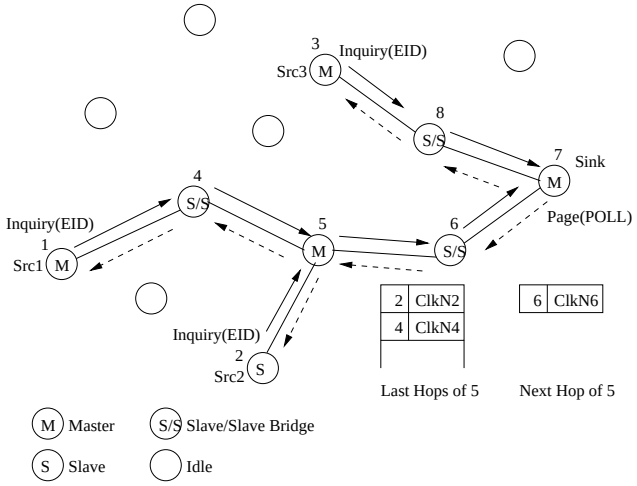


Figure 11. Scatternet and route formation

Instead of running a routing protocol after a scatternet is formed, the proposed approach combines these two processes. An overview of the protocol is shown in Figure 11. The route requests propagate through an *Inquiry* broadcast and are relayed from sources to the sink, while the scatternet formation and route replies are delivered in the opposite direction by *Page* messages. Since the common destination is the sink node, all route requests arriving at the same intermediate node (e.g., route requests from Src1 and Src2 arrive at node 5 in Figure 11) are merged, which avoids redundant request transmissions. It also makes nodes on cross routes join the same piconets as often as possible (e.g., node 2 and 4 join the same piconet). In addition, the support for concurrent cross route formation makes data aggregation easier, which is important for WSNs. All intermediate nodes buffer their last hop nodes' device addresses and clock values to *Page* them when route replies come back. After a Bluetooth device discovers a route to the sink, the next hop information is cached for a period of time. The timeout value for the cached route should be a function of network mobility. If new route requests arrive before the cached route to the sink expires, the next hop node will be paged first. The neighbor information cached at node 5 is also shown in Figure 11. The selection of forwarding nodes for route requests is based on the

residual energy of each node receiving the requests. The details of scatternet formation and route discovery are discussed in the following sections.

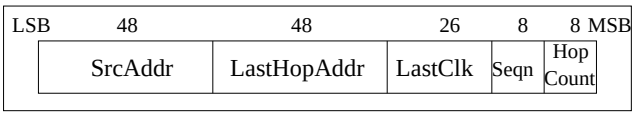
2.3.2 Route Request

2.3.2.1 Extended ID Packet and Modified Inquiry

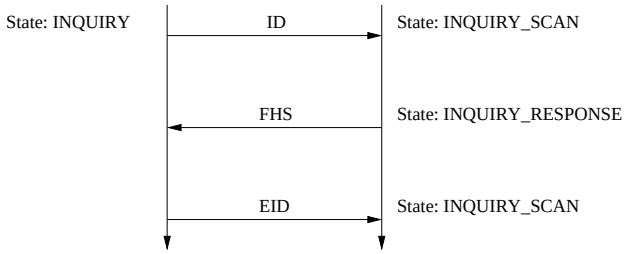
To propagate source information in the downstream direction from the source to the sink during the route discovery phase, an extended ID packet (EID) is used. It includes the *Inquiry* master's address and clock as well as other route request information. EID packets in the proposed protocol are used in the modified *Inquiry* rather than replacing the original ID packets as in [27, 29]. In the modified *Inquiry*, after the master discovers new devices, it sends an EID packet to transfer route requests to the newly discovered devices.

Each field and its corresponding length in an EID packet is shown in Figure 12(a). The *SrcAddr* is the Bluetooth device address of the node initiating the scatterent and route formation. The fields of *LastHopAddr* and *LastClk* denote Bluetooth device address and native clock of the immediate upstream node, respectively. They are useful for route reply transmission. *Seqn* is used to distinguish old and new requests from the same source, while *HopCount* limits the number of hops a request traverses. The *Inquiry* process in the proposed protocol is modified to accommodate the new EID packet. The modified *Inquiry* process is illustrated in Figure 12(b). An EID packet is sent by the upstream node after a new device discovery. In this case, the downstream node can get the source and last hop information from the *Inquiry* process, while normal small ID packets are still used for the neighbor and synchronization probes. Upon receiving the EID packet, intermediate nodes function as relays.

The modified *Inquiry* with EID packets solves the asymmetric device discovery problem in Bluetooth. Although a symmetric link formation protocol in [34] tried to solve this problem by alternating master/slave roles, the knowledge exchanged in *Inquiry* process is still not reciprocal. Devices in *Inquiry* states have knowledge of devices in *Inquiry Scan*



(a) EID packet format



(b) Modified Inquiry process

Figure 12. Modified Inquiry with EID packet

states, but not vice versa. One other work that tries to get mutual knowledge for both master and slave is presented in [35]. In that work, a temporary piconet is set up once a new neighbor is discovered. This piconet is transient, lasting only long enough for the exchange of device address, clock value, and other relevant information to achieve mutual knowledge. Although mutual knowledge is attained by the scheme, both master and slave need to leave *Inquiry* and *Inquiry Scan* states to set up a temporary piconet by the *Page* process. During this period of piconet setup, a master cannot discover new neighbors and a slave cannot be discovered by other masters. In addition, frequent state switching is needed for the temporary piconet setup. On the contrary, the proposed modified *Inquiry* with Extended ID packets in this research does not need to set up a temporary piconet and switch between *Inquiry* and *Page* regularly. Moreover, without leaving *Inquiry* or *Inquiry Scan* states, the Bluetooth device discovery process is accelerated.

Another problem for Bluetooth device discovery is the discovery duration. Most of the existing schemes rely on timeouts such as in [25]. A recent study on the number of neighbors needed to form a connected wireless network is presented in [36]. It is proved that the network is asymptotically connected if each node is connected to a set of neighbors

and the number of neighbors is larger than a value related to the total number of nodes in the network. In the Bluetooth specification, when more than seven slaves exist within a piconet, some of them must be parked, which is complicated and time consuming. It is shown in [37] through extensive simulations that in Bluetooth scatternet six and seven neighbors can guarantee the topology connectivity with high probability. Therefore, the device discovery can be terminated when the specific number of neighbors has been detected. Exiting device discovery by exploiting the limit on the discovered neighbors, rather than waiting for a timeout, has twofold benefits. First, the duration for device discovery is reduced without waiting for timeouts. Second, the number of slaves within a piconet is never more than seven. As a result, no device parking is needed in the scatternet formation. The neighbor limit scheme is employed in this research.

2.3.2.2 Factors Affecting Bluetooth Device Discovery

In [38], the connection establishment of two Bluetooth devices is evaluated. However, this process becomes complicated when multiple devices exist and interfere with each other, which we will evaluate in the following sections.

In order to measure the speed of device discovery when multiple Bluetooth devices coexist, we limited the number of neighbors to no more than 7 as discussed in the above section. The average node degree was used to indicate the effect of device discovery. The closer the average node degree to the discovered neighbor limit, the higher the probability that the discovered topology is connected. At the same time, the 7 discovered neighbor limit also guarantees that no more than 7 slaves can exist within one piconet. The scatternet formation and communication is simplified thereafter without parking slaves.

To evaluate the effects of various parameters' on device discovery latency, we ran simulation experiments using *GTNetS*. In order to investigate the properties of device discovery for multiple Bluetooth devices to form a scatternet, we enhanced the simulator with mechanisms for each node to assume a master or slave role with dynamic probability. In addition, the state alternation between *Inquiry* and *Inquiry Scan* was implemented for the symmetric

device discovery.

We ran the simulations following the specification of Class 3 Bluetooth devices with a radio range of 10 meters. The Bluetooth nodes were randomly and uniformly distributed in a geographic square area with sides of 30 meters. The number of Bluetooth nodes was set at 50 to evaluate the performance under moderate node density. The number of neighbors was limited to 7. The effects of the residence time and the probability for a node to assume master role initially are discussed next.

Effect of Residence Time In order for Bluetooth devices without preassigned roles to discover each other, alternation between *Inquiry* and *Inquiry Scan* is needed. Due to the frequency synchronization delay, the residence time at each state of the alternation has an effect on the device discovery delay. It has been shown that fixed state residence time results in arbitrarily large connection establishment time [39]. So we chose random residence time with uniform distribution and varied its mean value. A node was assumed to be a master or a slave role initially with equal probability. We ran the device discovery for 10 seconds and 20 seconds respectively and collected the average node degree. The simulation results are shown in Figure 13.

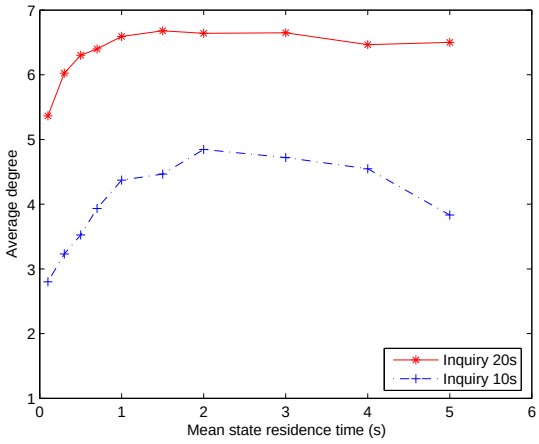


Figure 13. Effect of mean residence time

Figure 13 shows that the average node degree is relatively stable for the 20 seconds device discovery period as long as the mean residence time is larger than 1 second. This

indicates that 20 seconds is sufficient time for device discovery. The average node degree is very close to 7, which means the discovered neighbors can guarantee excellent connectivity of the topology. When the device discovery period is reduced to 10 seconds, the average node degree decreases when the mean residence time is less than 1 second and larger than 4 seconds. This is because when the residence time is too small, it takes several state switches in order to finish the *Inquiry* process, which includes the frequency synchronization, backoff, and frequency synchronization again. Therefore, the number of detected neighbors is lower within a specific time duration. The reason for the decreased average node degree with large residence time is waiting in vain for a state switch when devices are at the same states instead of the opposite *Inquiry* and *Inquiry Scan* states. Figure 13 shows that the optimal mean residence time is around 2 seconds. Thus, the device discovery time can be improved by appropriately choosing mean residence time.

Effect of Initial State The speed at which Bluetooth devices without preassigned roles can discover their neighbors depends on the state switch frequency of each device as discussed in the above section. Does it also depend on the initial roles that the devices enter? In order to answer this question, we varied the probability that a device is initially assigned as master, and measured the average node degree. The mean state residence time was chosen to be 2 seconds as an optimal value resulting from the residence time experiments. The simulation results are shown in Figure 14.

From Figure 14 we can see that the probability of a device being initially assigned as master doesn't affect the performance of device discovery too much for the *Inquiry* period of 20 seconds. When the device discovery period is 10 seconds, the average node degree is a little higher with medium probability of assuming master role initially. However, the differences are very small. This is different from our intuition that lower probability of being assigned as master will result in fast device discovery because of the master/slave ratio within a piconet. The constant performance for various probability indicates that the possibility of discovering neighbors with initial state settings is very small. Most of the

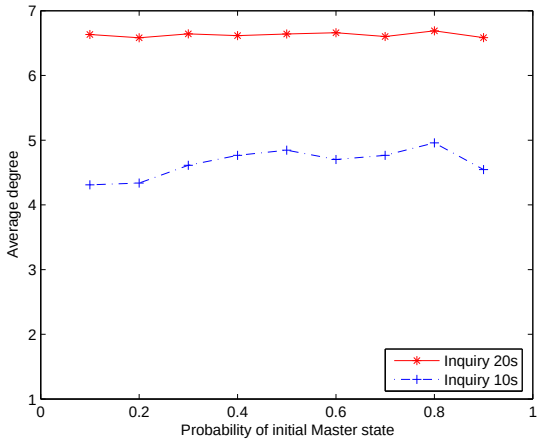


Figure 14. Effect of probability of initial master state

device discovery actions require state switches. So the residence time plays a much more important role in Bluetooth device discovery.

2.3.2.3 Route Request Forwarding

The pseudocode of route request flooding in the forward direction of scatternet and route formation from the source to the sink is given in Figure 15. When a source has traffic to send to the sink, it starts a scatternet and route formation using *Inquiry* to search bridge nodes to reach the sink. The modified *Inquiry* process shown in Figure 12(b) is being used. A scatternet formation timer *ScatFormTO* is started. This timer is stopped as soon as the scatternet formation and route reply arrives at the source. The intermediate nodes receiving route requests save the information about the source and last hop in a structure, *PrecursorList*, which is used to relay route reply to the upstream nodes as well as avoid flooding loops.

The intermediate nodes receiving route requests initiate their own *Inquiry* process to probe the next hop nodes toward the sink. At *Inquiry* timeout, the nodes that have detected their next hops switch between *Page_Scan* and *Inquiry_Scan* states. The state switch enables the nodes to wait for the scatternet and route formation replies in *Page* messages from the downstream nodes. At the same time, the nodes in route discovery can still accept new

```
Route_Request_Propagate()
  Initialize();
    Source: State = INQUIRY, ScatType = SOURCE;
    Dest: State = INQUIRY_SCAN, ScatType = DEST;
    Others: State = INQUIRY_SCAN, ScatType = IDLE;
  Start ScatFormTO;
  RecvEIDPacket();
    switch(ScatType)
      case IDLE:     //first recv EID packet
        Save EID info in PrecursorList;
        ScatType = BRIDGE;
        Inquiry();  //forward route request
        INQUIRY_SCAN/PAGE_SCAN switch on Inquiry timeout;
      case BRIDGE:     //already in route discovery
        if(new source request)
          Save EID info in PrecursorList;
        else if(existing source with smaller HopCount)
          Update PrecursorList;
        else if(existing source with larger Seqn)
          Update PrecursorList;
          Inquiry();
      case DEST:     //arrive at dest
        if(new source request)
          Save EID info in PrecursorList;
        else if(existing source with smaller HopCount or larger Seqn)
          Update PrecursorList;
        Route_Reply_Propagate();
    end;
end;
```

Figure 15. Pseudocode of route request forwarding

route requests from other sources. This enables multiple sources to start route requests simultaneously. The period of the state switch affects the performance in terms of scatternet formation delay, which is discussed in Section 2.4.3.

When route requests arrive at a node that is already in route discovery, the information from the new source requests will be saved in the *PrecursorList*. However, no *Inquiry* messages are generated. If a route request arrives at a node from a source already in the node's *PrecursorList* but traverses through a different route, the entry in the *PrecursorList* is updated to the route with the shorter path. For a route request with the same source address but larger *Seqn*, a new *Inquiry* message is generated since the previous request will get no response as a result of packet loss or *Inquiry/Page* failure.

After the scatternet and route formation requests arrive at a sink, the route reply will propagate in the reverse direction of the route request and the scatternet will be formed hop by hop. There is a delay between the first route request arrival at the sink and the initiation of route reply and scatternet formation. This short delay enables multiple requests to arrive at the sink and share the same scatternet formation using the sink's immediate hop. In addition, during this delay period, requests from the same source but via different paths can arrive at the sink. In this case, the route with smaller *HopCount* will be chosen, which decreases the number of piconets on a route.

2.3.3 Scatternet Formation and Route Reply

Upon receiving the route request and after a short delay, the sink responds with a route reply using a *Page* message and initiates the scatternet formation process.

2.3.3.1 Modified Page

In the proposed on–demand scatternet formation approach, a strict master/slave role alternation is maintained along any route from a source to a sink. To transfer the scatternet role assignment information along a route without extra overhead, a *Page* procedure with modified *POLL* packets is used.

In the Bluetooth *Page* process, a master assigns a non–zero active member address (*AMAddr*) to each slave within its piconet through the *POLL* packets. In a scatternet with alternating master/slave roles (*ScatRole*), the *AMAddr* assigned by the nodes with *ScatRole* set to slave has no meaning since the *Page* from these nodes is only for scatternet formation and route reply information propagation. In addition, broadcast packets with *AMAddr* of zero are not used in the scatternet connection–establishing phase. When a node with a *ScatRole* of slave tries to page its last hop nodes to relay the scatternet formation information, the *AMAddr* in the *POLL* packet is set to zero, while the *AMAddr* assigned by nodes with *ScatRole* of master is between one and seven. In this case, the upstream nodes in the scatternet route can decide their *ScatRoles* based on the *AMAddr*. With the modified *Page* scheme, *ScatRole* information is transfered without overhead and there is no effect on the proper operation of the scatternet formation.

2.3.3.2 Route Reply Propagation and Scatternet Formation

The scatternet formation is initiated by the sink. The sink sets its *ScatRole* to master and forms its piconet by paging all the last hop nodes in its *PrecursorList* to establish connection channels. The pseudocode of the scatternet formation and route reply propagation is given in Figure 16. The intermediate nodes being paged get their next hop's address and clock value through the *Page* process, and then page their own upstream nodes. For nodes with *ScatRole* of slave, they only page their upstream nodes to transfer scatternet formation and route reply information and then switch to *Page_Scan* state to wait for participating piconets of their upstream nodes. On the other hand, the nodes with *ScatRole* of master page both their last hop nodes and next hop node to form their own piconets.

To avoid multiple nodes with *ScatRole* of master paging the same next hop node simultaneously, a random backoff is used. The scatternet formation for these masters starts when the backoff timer expires.

```
Route_Reply_Propagate()
   Dest: ScatRole = Master, Page(all nodes in PrecursorList);
   BRIDGE and SOURCE:
     RecvFHSPacket();
       Get NextHop, NextClk from FHS packet;
     RecvPOLLPacket();
       if(AMAddr = 0) //next hop is slave
         ScatRole = Master;
       else //next hop is master
         ScatRole = Slave;
       SendNULLPacket();
       if(first POLL packet for scatternet formation)
         Random backoff to schedule StartScatForm();
     RecvNULLPacket();
       if(more last hop/next hop nodes need page)
         Page(last hop/next hop nodes);
       else if(ScatRole = Slave)
         State = PAGE_SCAN;
     Timeout for random backoff: StartScatForm();
       StartScatForm();
         if(ScatRole = Master)
           Page(all nodes in PrecursorList and NextHop);
         else if(ScatRole = Slave)
           Page(all nodes in PrecursorList with AMAddr = 0);
end;
(AMAddr: Active Member Address)
```

Figure 16. Pseudocode of scatternet formation and route reply

2.3.4 Energy–Aware Scatternet Formation

In wireless sensor networks, the energy consumption of each node is important because of the limited power provided by batteries. A large number of sensor nodes deployed in neighboring terrains form a dense multi–hop ad–hoc network, in which every node can work as a router to forward data to the sink. Therefore, how to balance the traffic load over the entire network and prolong the lifetime through efficient scatternet formation and routing is one of the primary concerns for WSNs. We investigate the effect of our energy–aware scatternet formation protocol in prolonging network lifetime with different network lifetime definition as discussed in [40].

The new protocol for scatternet formation proposed in this research is energy–aware in that the neighboring nodes with more power are selected preferentially over neighbors with less power. It has been observed that the sensor mote's battery life linearly declines with the current consumption[41]. Hence, to extend network lifetime, the selection of relaying nodes from the sensor sources to the sink should make use of the information of residual current in the intermediate nodes.

In this on–demand scatternet formation and routing protocol, the scatternet is formed depending on traffic patterns and no neighbor information between sensor nodes is exchanged. Therefore, when the scatternet and route request arrives at an intermediate node, the node makes the decision of whether to forward it or not based on its own residual current level. Since prolonging the network lifetime results from delaying some nodes to be energy depleted much earlier than the others, the objective of this protocol is to keep the energy consumption of all the nodes within the network at approximately the same rate and to avoid extra load at any given nodes. A node with the residual current (C_R) satisfying Equation (3) stops forward route requests for other sensor nodes.

$$C_R < [C_B - \frac{T_{Active}}{2 * T_{Slot}} * (C_{MaxTx} + C_{MaxRx})]\ \alpha \tag{3}$$

where C_B is the battery current, T_{Active} is device active time, T_{Slot} is Bluetooth slot time,

C_{MaxTx} and C_{MaxRx} are maximum transmission and reception current respectively, and α is the current factor.

If a node continually transmits and receives data packets during its active time, its residual current will equal the right side of Equation (3) (α=1). However, during the connection establishment phase, most of the packets being transmitted and received are small size control packets. Thus, for the same active time, the residual current should be larger than the extreme condition. In this case, a current factor α is introduced. The tuning of α balances the traffic load throughout the network. When α equals one, every node's residual current is greater than the extreme condition. Therefore, no traffic balance is achieved. With α greater than one, some intermediate sensor nodes with heavy forwarding traffic will satisfy Equation (3) at some time, and thus other nodes with large residual current will take on the load. Since our scatternet and route formation is initiated in an on–demand manner, the nodes with faster current consumption previously may become available after some recovery time and assume forwarding responsibility again when new requests arrive. From our experiments, the recovery time for an intermediate node to resume forwarding packets is in the order of 10s of seconds with the reasonable setting of the current factor. The tuning of the current factor is discussed in Section 2.4.4.

2.4 Performance Evaluation

In this section, a quantitative evaluation of the proposed on–demand scatternet formation and routing protocol is provided by means of analysis and simulation.

2.4.1 EID Power Saving

The power consumption resulting from the transmission of an EID packet is much more than that of smaller ID packets. Therefore, reducing the number of EID packets and transmitting them only when they are necessary is essential for power saving. Compared to [27] and [29], which also introduce types of EID packets and substitute EID packets for ID packets in all cases, we retain the smaller ID packets for neighbor probing in *Inquiry*

and only transmit EID packets when the source and last hop information is necessary for scatternet formation by the modified *Inquiry* process.

The power savings of the proposed *Inquiry* scheme is related to the period of *Inquiry* and the number of nodes participating the scatternet formation in the network.

According to the Bluetooth specification [2], the master must stay in the *Inquiry* state for 10.24 seconds to collect sufficient responses from its neighbors. However, the time to get enough responses varies significantly, depending on the alignments of device clocks. Simulations reveal that 5 seconds is sufficient most of the time for devices with preassigned master/slave roles.

The current consumption [2] comparison of the modified *Inquiry* to the traditional *Inquiry* with EID packets for one node is shown in Figure 17. The current consumption goes up linearly as the time for *Inquiry* increases, as a result of more EID packets being transmitted. However, the current consumption of our scheme achieves substantial savings. With *Inquiry* time of 10.24 seconds, 33.41% current saving is achieved by the modified *Inquiry* scheme. In wireless sensor networks with a large number of nodes, this saving is significant.

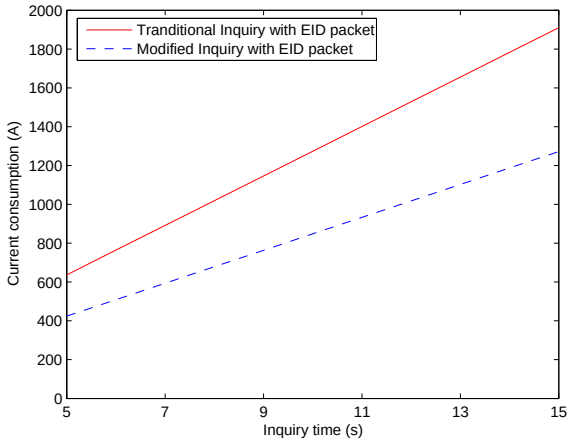


Figure 17. Current consumption of Inquiry

[2]The current drain for a typical Bluetooth device to transmit an ID packet is 26.5mA and it is 39.8mA for an EID packet [42].

2.4.2 Accelerating Device Discovery

To measure the speedup of device discovery using the modified *Inquiry* with EID packets when multiple Bluetooth devices exist within radio range, the number of neighbors was limited to no more than seven, as it is shown in [37] that seven discovered neighbors can guarantee the connectivity of the topology with high probability. The average node degree was used to indicate the effect of device discovery. The closer the average node degree to the discovered neighbor limit, the higher the probability that the discovered topology is connected. At the same time, the seven discovered neighbor limit also guarantees that no more than seven slaves can exist within one piconet. The scatternet formation and communication are simplified thereafter without parking slaves.

The protocol is implemented and simulated in *Georgia Tech Network Simulator* (*GT-NetS*) [31, 32]. The performance of the modified *Inquiry* with EID packets is compared to the traditional *Inquiry* process.

The simulations were run following the specification of Class 3 Bluetooth devices with a radio range of 10 meters. The Bluetooth nodes were randomly and uniformly distributed in a geographic square area with sides of 30 meters. The number of Bluetooth nodes uniformly distributed in the area was 30, 50, 70, 90, and 110, respectively, to evaluate the performance under different node densities from sparse networks to dense networks. The probability of the initial role as master was 0.5, and the mean residence time was 2 seconds as the optimal selection from the former evaluation [43]. Figure 18 illustrates the simulation results from these experiments.

It is shown in Figure 18 that the average node degree increases for the same device discovery period as the node density increases. This is obvious since more neighbors exist in dense networks, leading to more chances for frequency matching. When the device discovery period is 20 seconds, the average node degree difference between the traditional *Inquiry* and the modified *Inquiry* with EID packets is small. This is because a 20-second discovery period gives more than enough time for neighbor detection even for the traditional *Inquiry*.

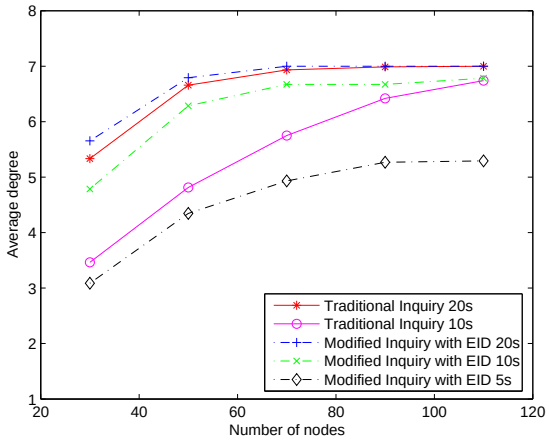


Figure 18. Comparison of average degree

To distinguish the promptness of device discovery, the discovery period was reduced to 10 seconds. In this case, the modified *Inquiry* scheme shows an advantage over the traditional *Inquiry*, especially for a sparse network without high node density. The performance of the modified *Inquiry* scheme with a 10-second discovery period is comparable to the traditional *Inquiry* scheme with a 20-second discovery period. This is due to the mutual knowledge exchange introduced in the EID packets. In addition, although the average node degree decreases with low node density, the modified *Inquiry* scheme achieves a much flatter curve as the node density decreases. Even with the 30-node case, the average node degree for the modified *Inquiry* is still 4.79, whereas it is only 3.46 for the traditional *Inquiry* when the discovery period is 10 seconds. As the discovery period further reduces to 5 seconds, the average degree of the modified *Inquiry* for a highly dense network (110 nodes) is only 5.29. Therefore, 10 seconds is a good point for the balance of quick device discovery and topology connectivity guarantee for Bluetooth devices without preassigned roles.

2.4.3 Scatternet Formation Delay

The intention of choosing an on–demand rather than a proactive scatternet formation scheme is to conserve power used by connection maintenance for the entire network. However, the

trade–off of on–demand approaches versus proactive approaches is the scatternet formation delay incurred by on–demand methods. Therefore, the scatternet formation delay is an important metric. To measure the scatternet formation delay, we implemented our scatternet formation protocol in *GTNetS*. The network topology chosen for the simulations is shown in Figure 19. This grid topology has one sink and multiple sources. Every node not residing on edges has eight neighbors within its radio range. This topology is similar to a typical cluster of monitoring or data collection sensor network applications with all possible source distributions relative to the sink. The network topology can be extended by connecting multiple clusters to form a larger network.

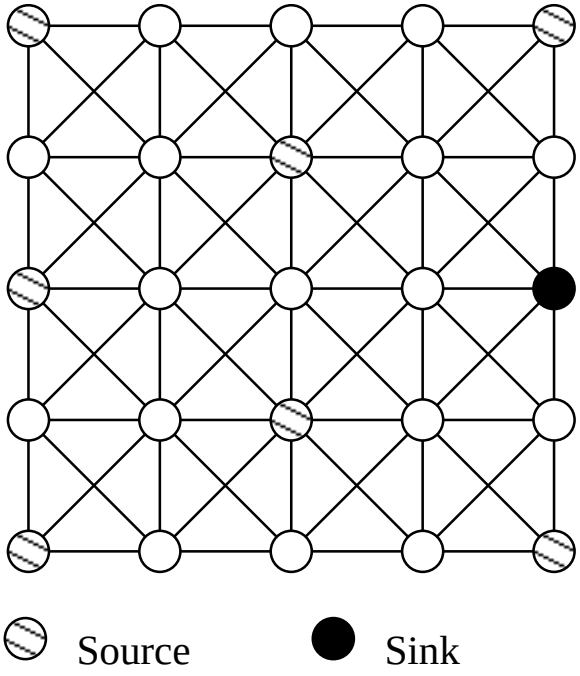


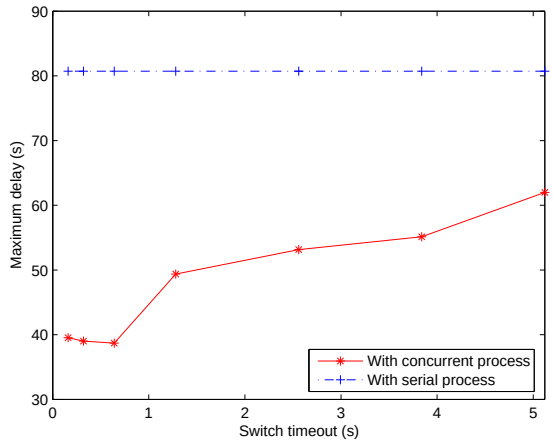
Figure 19. Multihop network topology

As stated in Section 2.3.2.3, the value of the switch timeout (*SwitchTO*), which controls the alternation of *Inquiry_Scan* and *Page_Scan* states after routing request forwarding, has a significant effect on the scatternet formation delay. In Figure 20, the *SwitchTO* is varied from 0.16 seconds to 5.12 seconds to measure the maximum as well as the total delay for all the sources to finish the scatternet formation process. Figure 20(a) shows that the maximum scatternet formation delay with our simultaneous process for cross routes is greatly reduced

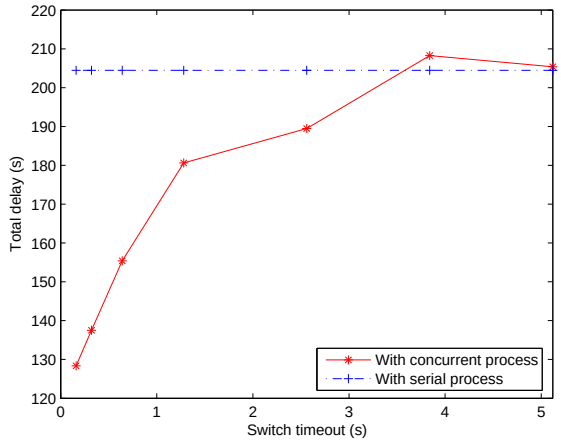
compared to the serial formation process. The *SwitchTO* value of 0.64 seconds achieves the best performance for the maximum scatternet formation delay, which is only 47.94% of the delay with the serial process. The optimal *SwitchTO* value occurs at 0.64 seconds. This is because when the timeout value is too small, the node switches frequently between the two states (*Inquiry_Scan* and *Page_Scan*) and stays a short time in each state. Since Bluetooth uses frequency hopping spread spectrum, the switch interval is not enough for the *Inquiry* or *Page* nodes to hop to the frequencies being listened to by the scanning nodes. If the timeout value is too large, time is wasted waiting for state switch. This optimal *SwitchTO* is also coincident with the average *Page* delay, which is half of the *Page_Scan* window (1.28s). The page hopping sequence is determined by the device address and clock of the paged device. Since this information is obtained by the downstream node from the EID packet in the route request phase, the downstream node can estimate the current *PAGE_SCAN* frequencies of its upstream nodes and reduce the page delay to less than one *PAGE_SCAN* window (1.28s). The average page delay is 0.64 seconds (1.28s/2). That is the reason the optimal maximum scatternet formation delay occurs at 0.64 seconds.

While the maximum scatternet formation delay measures how long it takes all the sources starting traffic simultaneously to finish the sctternet formation, the total delay gives the sum of the formation delay of each source. The total scatternet formation delay of all the sources is illustrated in Figure 20(b). The delay with concurrent process is lower than the case with serial process (75.99% with *SwitchTO* of 0.64s) when the *SwitchTO* value is less than 2.56 seconds, which is the value for *Page* timeout. The optimal *SwitchTO* is no longer 0.64 seconds. This is because some sources with a small number of hops to the sink sacrifice their own formation delays, which increase the total formation delay, but benefit the nodes with a large number of hops from the sink to complete the scatternet formation process promptly.

Another parameter affecting the performance of the scatternet formation, which must be tuned carefully, is the timeout value for the scatternet formation (*ScatFormTO*). The



(a) Maximum scatternet formation delay



(b) Total scatternet formation delay

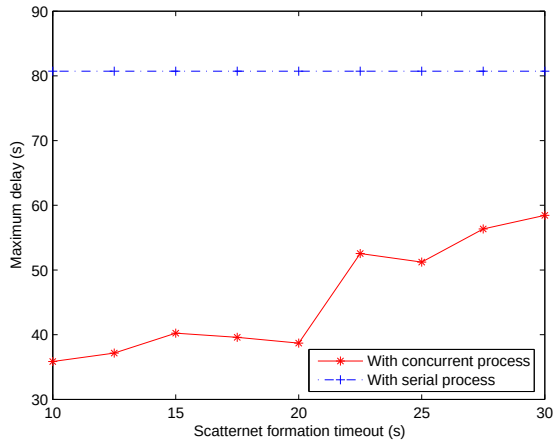
Figure 20. Scatternet formation delay vs. switch timeout

source nodes initiate new scatternet formation requests when no responses arrive and *ScatFormTOs* expire. If the *ScatFormTO* is set to be too small, new requests are sent out before normal replies come back. Thus, the scatternet formation is initiated repeatedly without any success. On the other hand, setting *ScatFormTO* too large may incur unnecessary delay to wait for the scatternet formation timeout and recover from failures. We varied the *ScatFormTO* from 10 seconds to 30 seconds to look for the optimal selection. It can be observed from Figure 21 that the delays remain stable when the *ScatFormTO* is less than 20 seconds and increase significantly with the *ScatFormTO* larger than 20 seconds. This is due to the time spent waiting in vain for a reply.

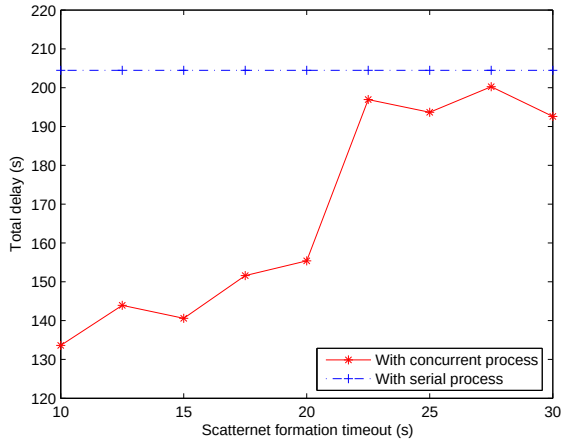
2.4.4 Prolonging Network Lifetime

One of the benefits of on–demand scatternet formation and routing schemes compared to proactive methods is that traffic can be routed through different paths every time a new request is initiated. Therefore, the traffic load can be easily distributed to balance the energy consumption of each node and decelerate energy depletion of nodes lying along overloaded paths. The lifetimes of the network with and without an energy–aware scheme were compared under the same traffic patterns as above. We first define network lifetime as the first node in the network is depleted of its energy. After that, we investigate the effect by introducing the connectivity network lifetime as defined in [40]. Simulation results for various numbers of simultaneous sources are shown in Figure 22 and Figure 23 respectively.

As discussed in Section 2.3.4, the tuning of the current factor α can balance the traffic in the network. The α was varied from 1 to 1.8 to search for the best performance of resource utilization. When the α equals 1, the network lifetime with load balance has no effect and shares the same performance as the normal scheme. This is because the value of $\alpha = 1$ is an extreme condition that all nodes can pass the evaluation of Equation (3) and act as forwarding nodes for scatternet and route formation requests. When α is greater than one, after some nodes participate in the formed scatternet and data relay, their residual current decreases. As new requests arrive at these nodes, they keep silent if their residual current



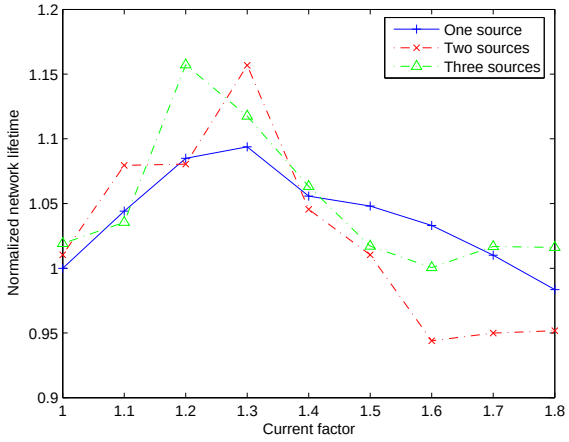
(a) Maximum scatternet formation delay



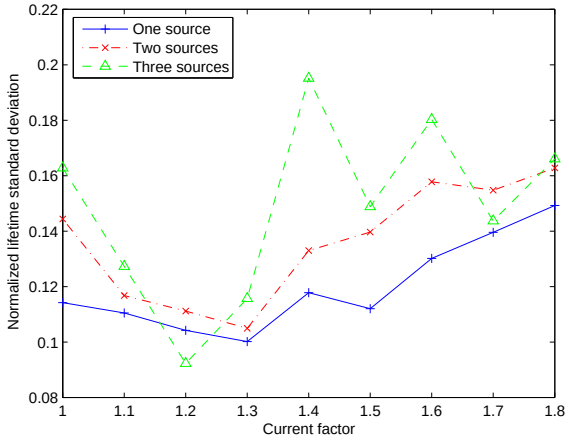
(b) Total scatternet formation delay

Figure 21. Scatternet formation delay vs. scatternet formation timeout

satisfies Equation (3). In this case, other nodes with high residual current will accept the requests and forward the traffic. The estimation of the nodes' residual current changes with time. Therefore, all the nodes in the network consume energy at approximately the same rate and the network lifetime is extended.



(a) Network lifetime



(b) Network lifetime deviation

Figure 22. Network lifetime comparison with/without load balance

From Figure 22(a), it can be observed that the network lifetime first increases with the current factor and then decreases as the current factor goes up. The increase is due to the load balance discussed before. When the current factor becomes too large, no node

considers itself capable of supporting the present scatternet formation and route request. Thus a lot of energy is drained by the transmission of scatternet request control packets until some nodes can accept the requests as time goes on. The standard deviation is in the opposite trend of the lifetime since lower deviation means a higher degree of load balance and hence longer lifetime.

In [40], the network lifetime is referred as the length of the time that the network operates prior to becoming unusable. Based on this, a formal definition of network lifetime is given. Here, we choose the network lifetime definition in [40] that takes the network connectivity into consideration. We refer it as connectivity network lifetime hereafter. The comparison of the connectivity network lifetime for the scatternet formation with and without load balance is shown in Figure 23.

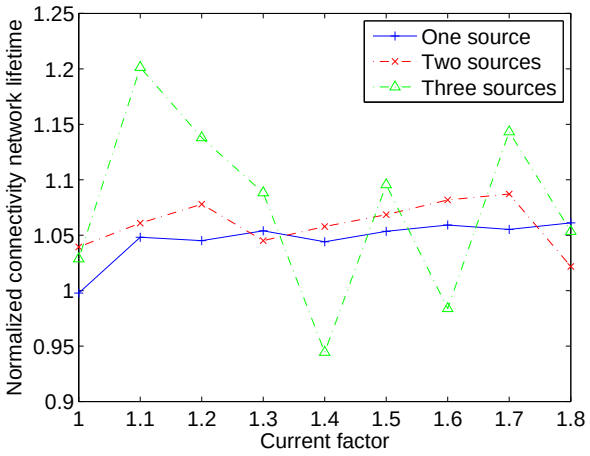


Figure 23. Connectivity network lifetime comparison

From Figure 23, we can see that for the one and two source scenarios, our energy–aware scheme gains a constant improvement of network lifetime in terms of network connectivity. When the number of sources increases, the improvement of network lifetime fluctuates. This is due to the sharing of some common intermediate nodes, which makes the lifetime more topology dependent. When the number of simultaneous sources increases, our forwarding node selection algorithm is expected to achieve optimal performance with current factor closer to one.

2.5 Conclusions

We have described an on–demand scatternet formation and routing protocol used for Bluetooth–based wireless sensor network applications. We deal with the complicated problem of supporting multiple sources that initiate the scatternet and route formation involving common Bluetooth devices at the same time. In addition, we modified the Bluetooth *Inquiry* process with extended ID (EID) packets for scatternet formation and route request propagation. We showed the power efficiency of this scheme comparing with traditional *Inquiry* with EID packets. Further, we employed a mechanism using *POLL* packets in *Page* mode to transfer scatternet formation information without extra expense. The energy aware scatternet formation and routing property makes the scheme even attractive by prolonging the network lifetime, which is a primary concern for wireless sensor networks. Simulation results demonstrate that our protocol achieves significant improvement in scatternet formation delay when compared to serial scatternet formation for multiple sources with concurrent traffic. The load balance scheme extends the network lifetime based on local information. It meets the requirements of Bluetooth–based wireless sensor networks in terms of power efficiency due to our on–demand rather than proactive approach in scatternet formation. At the same time, the protocol does not incur large scatternet formation delay and achieves uniform resource utilization.

CHAPTER 3

METRIC-BASED SCATTERNET FORMATION AND RECOVERY OPTIMIZATION FOR INTEL MOTE

3.1 Introduction

In the experience gained from the former wireless sensor network experiments and deployments, it has been shown that the system aspects are far more complicated than anticipated [18, 19]. This raises the necessity to not only model and simulate WSNs, but also to test on real systems.

Wireless sensor networks combine processing, sensing, and communications into tiny embedded devices [47]. The recent research and development of wireless sensor network platforms has open a new era for the innovation of hardware and software technologies. Based on the typical operating characteristics, four classes of sensor node platforms can be categorized as follows:

- **Specialized Sensing Platform –** The Spec node [48] designed at the University of California, Berkeley, is representative of the special purpose sensing platform. The Spec node is built to be ultra low cost and low power. Therefore, it has very simple interface and short communication distance.

- **Generic Sensing Platform –** The Berkeley Motes, recently known as Mica2 [49, 50], are a notable example of generic sensing platform. With a large number of I/O interfaces and expansion options, the Mica2 is an option for many sensor network application. However, to process video or hight bandwidth audio coming from sensor, the Mica2 node is not so capable.

- **High Bandwidth Sensing Platform –** Intel Mote [15] is designed to be a high bandwidth sensor platform with a Bluetooth–based radio and large RAM. It can handle data rate larger than 500 Kbps.

- **Gateway Sensing Platform –** Most of the sensor networks have a hierarchical architecture with low–level sensors connecting to high–level gateways. The Stargate [51] platform developed by Intel is a representative of gateway class devices. It has more powerful capability in processing and large memory. The Stargate provides interfaces for multiple sensor platform such as Mica2 and Intel Mote and acts as bridge between low–power mesh networks to traditional networks (802.11, Ethernet, etc.).

The Intel Mote is an enhanced sensor network node platform which can provide applications with improved computation and communication abilities [15, 46]. It uses an integrated wireless micro–controller module that incorporates an ARM7TDMI core and a CMOS Bluetooth [2] radio. The adoption of Bluetooth radio inherits the merits of high data rate and reliable link, which are the requirements of some realistic industrial deployments [52].

3.1.1 Problem Description

Despite various advantages of Bluetooth as a short–range wireless network technology, the scatternet formation mechanisms have a great impact on the performance of the networks with Bluetooth devices. An analytical work in [53] shows the relationship between the average path length of a Bluetooth scatternet and the overall throughput and power consumption of the network. It shows that by reducing the hop count in a scatternet, better performance in terms of throughput and power consumption can be achieved. Therefore, reducing the average hop count is one of our considerations in the scatternet formation optimization for the Bluetooth–based wireless sensor networks.

The lossy nature of wireless communications, especially low–power radio for wireless sensor networks, is a major obstacle for network reliability. Assuming the links between pairs of nodes either work well or do not work at all is not realistic in wireless networks. So it is not enough to merely reduce the hop count within a scatternet. Another metric that impacts the Bluetooth scatternet formation is link quality (LQ), which is the indication of

the bit error rate (BER). Constructing a Bluetooth scatternet with high link quality (low BER) connections increases the network reliability and reduces network repair overhead.

In wireless sensor networks with dynamic link and node failures, the maintenance of the scatternet becomes an important issue. Most of the Bluetooth scatternet formation algorithms in WSNs, such as that found in the Intel Mote, forms a tree topology with a preset root node. In the case of a node or link failure, all the nodes within the subtree disconnect and they attempt to rejoin the main tree individually. This is an inefficient process from the power and delay perspectives, especially due to the long device discovery time in Bluetooth. Thus, a more intelligent recovery mechanism is required for network repair.

3.1.2 Related Work

One effort to investigate the robust topology formation and optimization using Bluetooth–base BTnodes is presented in [54]. In this work, a simple Bluetooth scatternet formation algorithm is implementation on real devices. A tree topology is constructed to connect multi–hop Bluetooth devices based on a search–and–connect scheme. Various aspects of Bluetooth distributed systems, such as simple design, robust and failure tolerant topology algorithms, are discussed in [54]. In this dissertation, we address enhancements to the non-deterministic Bluetooth scatternet formation behavior from the hop distance, link quality, and reliable perspectives.

3.1.3 Solution

Taking into consideration of all the issues in section 3.1.1, we design and evaluate a number of optimizations in the network formation and maintenance algorithms for the Bluetooth–based sensor motes (we take Intel Mote as an example mote platform). These optimizations include the metric–based scatternet formation by reducing hop distance and increasing link quality. They are easy to be incorporated into the existing mote implementation. These optimizations are efficient in improving the network throughput and reliability. To deal with

node/link failure for the Bluetooth–based mote constructed tree topology, an intelligent network repair scheme using Bluetooth low power hold mode is proposed. The experiments with the Intel Mote incorporating these optimizations demonstrate their effectiveness.

The details of the metric–based scatternet formation as well as the intelligent network recovery scheme are described as follows. Section 3.2 gives the detailed algorithms for the metric–based network formation, while Section 3.3 describes the optimized network maintenance mechanism. In Section 3.4, the extensive experiments with the optimizations in Intel Mote as well as analysis are presented. Finally, the conclusions are discussed in Section 3.5.

3.2 Metric–Based Scatternet Formation

The Intel Mote includes a scatternet formation algorithm which creates a network with a tree structure as described in [55]. One node is designated as the root node. Each node randomly enters either inquiry or inquiry scan state and alternates between these two states. If one node discovers another node, a connection is established and a synchronization message is exchanged which includes a flag to indicate whether this node is already a member of the tree (tree node). The connection is torn down if neither of these two nodes are tree nodes. Otherwise, the free node joins the tree by connecting to the tree node. The existing member of the tree becomes the master in the new piconet, while the originally free node becomes the slave. After a node joins the tree, it remains in inquiry scan state.

In the above algorithm, a free node connects to the tree via the first tree node it discovers regardless of the hop distance to the root node and the link quality. This may lead to a Bluetooth scatternet with very long routes from the root node to the leaves. In addition, the scatternet with poor link quality connections incurs frequent packet loss and link failure. In the following sections, the optimizations with metric–based scatternet formation are discussed.

3.2.1 Hop Distance–Based Formation

As compared to broadcast wireless technologies, the effect of the hop distance on a Bluetooth scatternet is more severe. The increase in hop count in a Bluetooth scatternet results in more piconet switches along the route, which further increases the forwarding traffic and delay. Hence, the reduction of the hop distance results in higher aggregate throughput and reduced power consumption [53]. Due to the frequency hopping nature of Bluetooth, the time to establish a connection between two nodes can be in the order of seconds. This makes an exhaustive shortest path search for such a network extremely time consuming. Therefore, we must find a balance between the hop distance and the network formation delay.

For resource constrained wireless sensor nodes, instead of computing and shifting to the optimal routes after the scatternet has been formed, we choose to limit the hop distance at the original scatternet formation phase to reduce connection re–establishment overhead. The hop distance–based scatternet formation algorithm is shown in Figure 24. The network formation process is controlled by the value of *Current_Hop_Threshold*. This threshold is initialized to be one, and incremented by the threshold timer as the process goes on. At the early stage of the scatternet formation, the hop distance to the root node is strictly restricted. This approach increases the probability of selecting the shortest path to the root node, as opposed to the first tree node responding to the inquiry. As the network formation process goes on, the hop distance restriction becomes looser with increased *Current_Hop_Threshold*. Finally, connections with any hop distance to the root node are accepted in order to limit the network formation delay.

In the hop distance–based scatternet formation algorithm, there is a tradeoff between the optimal hop distance and the network formation delay. So the selection of the threshold update interval is crucial, which will be discussed in Section 3.4.1.

```
Hop_Distance-Based_Formation()
  Initialize()
    Any_Hop_Distance = false;
    Current_Hop_Threshold = 1;
    Start_Threshold_Timer();
  Timer.fired()
    Current_Hop_Threshold += Hop_Increment;
    if (Current_Hop_Threshold > Threshold_Limit)
      Any_Hop_Distance = true;
  Network_Formation()
    if (This_Node == Root_Node)
      Accept any connection request;
    else if (Any_Hop_Distance)
          Accept any connection request;
        else if (Hop_Distance ≤ Current_Hop_Threshold)
            Accept the connection;
          else Disconnect the connection and retry;
end;
```

Figure 24. Hop distance–based scatternet formation algorithm

3.2.2 Link Quality–Base Formation

Because of the instability of wireless networks, routing algorithms for wireless sensor networks should take into account the underlying factors such as lossy links, interference, and congestion [56]. This cross layer design intends to improve the reliability of the formed scatternet. Due to the highly correlated behavior of sensor network applications and the dynamic environment, the communication links may not be reliable. Therefore, connectivity does not mean successful communication.

The Bluetooth MAC provides a primitive to retrieve the received signal strength (RSSI) and link quality indication (LQI). Both of these low level measurements can be accessed by Bluetooth host controller interface (HCI) through *Read_RSSI* and *Get_Link_Quality*. From our experiments with the Intel Mote and the experiments in [57, 58], there is a more accurate correlation between LQ and packet success rate than RSSI. The LQ in Bluetooth is an 8-bit unsigned integer in the range of 0 to 255, which is derived from the bit error rate (BER). The higher the LQ, the better quality of the link. The conversion of BER to LQ is

described in Equation (4).

$$BER = \begin{cases} 0, & LQ = 255 \\ (255 - LQ) * 0.0025, & 215 \leq LQ < 255 \\ 0.1 + (215 - LQ) * 0.08, & 90 \leq LQ < 215 \\ 10.1 + (90 - LQ) * 0.64, & 0 < LQ < 90 \\ 67.7, & LQ = 0 \end{cases} \quad (4)$$

The link quality observed is a result of averaging over a time period. The resolution of the link quality depends on the measurement update interval and the period over which the encountered BER is averaged. In our link quality–based scatternet formation algorithm, the link quality value is read right after the connection has been established. If the link quality does not meet the requirement, the connection is immediately torn down and retried. The accuracy of the link quality is improved by increasing the number of ACL packets exchanged during the averaging period. Since routing packets are exchanged right after a connection is established, we chose to sample the link quality indicator after the connection establishment, hence improving the LQI accuracy. This result has been verified in our experiments.

The link quality–based scatternet formation algorithm is shown in Figure 25. Similar to the hop distance–based scatternet formation scheme, this simple algorithm tries to look for the best quality link available to connect to the tree without requiring local storage of link quality information of the neighbors, hence reducing the memory requirements. With the link quality–based scatternet formation scheme, poor quality links with high error rate are avoided. In the case that there are only lossy links to the scatternet tree or the good quality link search incurs excessive long network formation delay, links with any LQ will be accepted after several connection attempts. This is necessary for the network connectivity and finite network formation time. Incorporating the link quality information into the scatternet formation improves the reliability of data communication and reduces network disconnections.

```
Link_Quality-Based_Formation()
   Initialize()
      Any_Link_Quality = false;
      Current_LQ_Threshold = 245;
      Start_Threshold_Timer();
   Timer.fired()
      Current_LQ_Threshold -= LQ_Decrement;
      if (Current_LQ_Threshold < Threshold_Limit)
        Any_Link_Quality = true;
   Network_Formation()
      Finish connection;
      if (Any_Link_Quality or LQ ≥ Current_LQ_Threshold)
        Go on with data communication;
      else if (LQ < Current_LQ_Threshold)
          Disconnect the connection and retry;
end;
```

Figure 25. Link quality–based scatternet formation algorithm

3.3 Intelligent Recovery

The topology constructed by the Intel Motes is a tree structure with a predefined root node. A simple routing algorithm is used to maintain paths between peer nodes. The root node periodically sends a routing beacon down the tree in order to establish and refresh the routing table. When a link or node fails, the routing beacon will not reach any of the downstream nodes. If a node receives a signal indicating that it loses its connection to its parent, and hence the connection to the root node, or the routing beacon is not heard for a period of time, it disconnects all its descendant nodes. The node detecting the failure and all its descendant nodes become free nodes and try to connect to the tree by going through the scatternet formation stage.

In the above repair mechanism, if a node closest to the root node fails and the number of its descendant nodes is large, such as the failed node ”A” shown in Figure 26, the repair time for all its descendant nodes to connect to the tree will be long and compared to the initial scatternet formation delay. Therefore, an intelligent recovery scheme is proposed as shown in Figure 26. The algorithm is depicted in Figure 27.

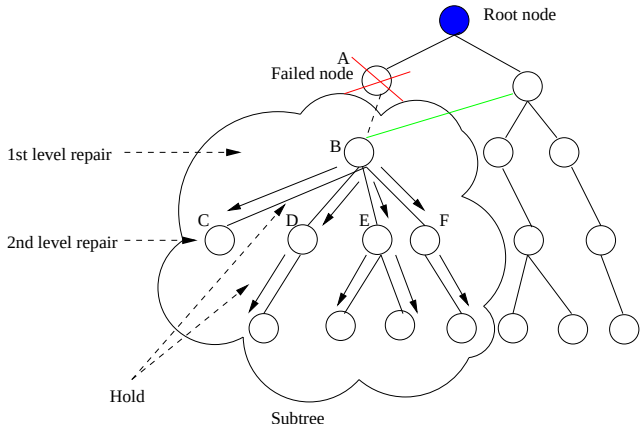


Figure 26. Intelligent recovery

In our recovery scheme, if a parent node detects losing connectivity to its parent, (node "B" in Figure 26), it initiates a *Repair()* process instead of disconnecting all its descendants. A repair timer is set with a timeout period of *Repair_Timeout*. All the connections to the descendant nodes are maintained. A hold command is sent through a reliable packet transport mechanism to all its descendants ordering them to stop scan. The repair node (node "B" in Figure 26) enters inquiry state and tries to connect to the tree. The reliable packet abstraction of TinyOS guarantees the hold message delivery. The nodes receiving the hold command will not respond to the repair node's inquiry, hence avoiding a routing loop. The repair node will become a free node if the repair does not succeed and the repair timer fires upon timeout. Note that repair timer for the hold nodes is a little longer than the repair node, which is *Repair_Timeout* + *Hold_Timeout*. This enables the multi–level repair if the first level repair fails. [1] Upon recovery by the repair node connecting to the tree, a routing refresh message is sent to all the descendant nodes in hold and the repair process finishes. If the upper level repair is unsuccessful on timeout, the repair node becomes free node and repair is initiated at the next level in the tree.

With our enhanced recovery algorithm, only the ancestor node detecting the link or node failure tries to connect to the tree. The node's subtree is maintained, and the connections are

[1] In Figure 26, the repair of node "B" is the first level repair. If node "B" repair fails, node "C" becomes free node while node "D", "E, "F" begin repair, which are the second level repairs.

```
Intelligent_Recovery()
   Disconnect() or Routing_Beacon_Timeout()
      if (This_Node != Root_Node and NumChildren != 0)
        Repair();
      else
        Become a free node;
   Repair()
      Beacon_Timer.stop();
      Repair_Timer.start(Repair_Timeout);
      Send_Repair_Hold();    //Keep descendant connections
      Inquiry();    //Go to Recovery() after connecting to tree node
   Recv_Repair_Hold()
      Stop inquiry scan;
      Beacon_Timer.stop();
      Repair_Timer.start(Repair_Timeout+Hold_Timeout);
      if (NumChildren != 0)
        Forward_Repair_Hold();
   Recovery()
      Repair_Timer.stop();
      Beacon_Timer.start();
      Send routing refresh to descendant nodes;
end;
```

Figure 27. Intelligent recovery algorithm

resumed once the repair is successful. If the repair process fails, the direct children of the node will attempt to initiate their own repair process. This process repeats until a leaf node is reached. Since the subtree is maintained, the recovery could result in lower overhead due to the reduction in control packet transmissions. Note that in a worst case scenario, the recovery time may be longer than disconnecting all the descendant nodes at the beginning of failure if the repair has to go down too many levels in the subtree. However, through the experiments in Section 3.4.3, the possibility of multi–level repair is small. Even including the recovery time for the multi–level repair, the average repair time is still much lower than the mechanism that disconnecting all the descendant nodes. We will show the comparison in detail in Section 3.4.3.

3.4 Experiments

We denote the network formation and recovery scheme in the current Intel Mote implementation as the original algorithm hereafter. We have measured the performance of our new algorithms using the Intel Mote in various settings and compared it with the original algorithms. The comparisons from the perspectives of hop distance, link quality, and network repair time are detailed in the following sections.

3.4.1 Hop Distance Effect

To evaluate the hop distance–based scatternet formation mechanism, we performed experiments with a 10 and 20 mote clusters. All the motes are within radio range of each other. The scatternet formation delay as well as the average hop distance from all the motes to the root node are collected. The results are averaged over 16 experiments.

In the case of the 10 mote cluster, the comparison of the hop distance distribution for the original algorithm and the hop distance–based scatternet formation scheme is shown in Figure 28. As we discussed in Section 3.2.1, the hop distance threshold is updated as the scatternet formation process goes on. The threshold update interval has an impact on balancing the average hop distance in the scatternet and the network formation delay. In

our experiments, we varied the threshold update interval from 30 seconds to 60 seconds. It can be seen from Figure 28 that the scatternet constructed without taking hop distance into consideration has the hop distance to the root node distributed across one to four hops for a 10 node network. In contrast, with 30 seconds hop distance threshold update interval, no node has more than 3 hops to the root node. By further increasing the threshold update interval to 60 seconds, the nodes have more time to search for shorter paths to the root node and discard long multi–hop routes. Therefore, most of the nodes are within one to two hops to the root node (5.13 nodes are one hop away, 4.8 nodes are two hops away). The reduced hop distance demonstrates the merit of our algorithm. As a result, the throughput and the power consumption will be improved as being proved in [53].

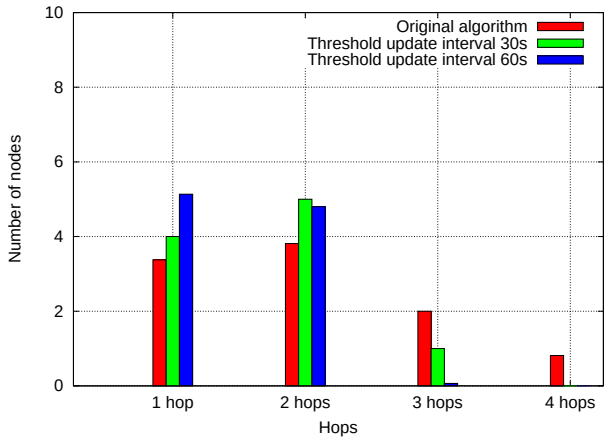


Figure 28. 10 nodes hop distance distribution

The tradeoff for the reduced hop distance in the formed scatternet is the increased scatternet formation delay. The average hop distance and scatternet formation delay are depicted in Table 2. For the 10 node network, the threshold update interval of 30 seconds decreases the average hop distance from 2.025 to 1.7. The cost for this is about 35 seconds increase in the scatternet formation delay. The larger the threshold update interval, the longer time for the nodes to search for shorter paths to the root node, hence the longer the network formation delay. The applications can choose different hop distance threshold update interval based on the specific requirement to balance the scatternet structure and the

network formation time. A point worth mentioning here is that in many industrial applications the formed topology will be maintained for a long time due to the static nature of nodes and the large amount of data that needs to be transfered across the network in a single data collection cycle. In such a usage model, the extra time spent in improving the network formation is negligible and more than compensated for by the reduction in the data transfer time due to the decrease in MAC level retries.

Table 2. Hop distance vs. formation delay (10 nodes)

Algorithm	Original	Threshold interval 30s	Threshold interval 60s
Avg hops	2.025	1.7	1.493
Avg delay(s)	52.975	87	131.313

To investigate the performance of the hop distance–based scatternet formation algorithm in larger networks, we increased the network size to 20 motes within radio range of each other. The hop distance distribution and the average network formation delay are shown in Figure 29 and Table 3 respectively. As the network size increases, the effect of our algorithm on the hop distance reduction is more significant. This is because in larger networks, there are more chances for nodes to form long multi–hop routes to the root node if hop distance is not taken into consideration. With the hop distance–based scatternet formation, almost all of the motes have no more than 3 hops to the root node, while the algorithm without hop distance consideration has nodes with hop distance up to 6. The average hop distance in the network decreased from 2.585 hops to 1.995 hops for every node, which is remarkable.

Table 3. Hop distance vs. formation delay (20 nodes)

Algorithm	Original	Threshold interval 30s
Avg hops	2.585	1.995
Avg delay(s)	62.53	148.81

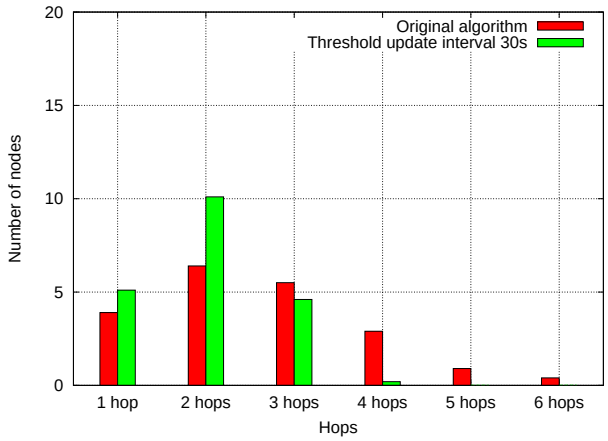


Figure 29. 20 nodes hop distance distribution

3.4.2 Link Quality Effect

Besides the hop distance, the link quality is another important metric in the consideration of reliable sensor network architecture. The experiments for the link quality–based scatternet formation are conducted outdoor with 10 nodes spread along a line. These nodes are not within the radio range of each other. So some of the links between distant nodes may have poor quality with high error rate. As a rule of thumb, a Bluetooth link with a BER between 0 and 0.1% is workable. Trying to run a link with a BER above 1% will give poor results. According to Equation (4), the workable link quality is in the range of [255, 215]. The link with quality lower than 204, which corresponds to 1% BER, has poor performance.

According to the above usable link quality, we set the link quality thresholds described in the link quality–based scatternet formation algorithm (Figure 25) with multiple levels from 255 to 204 and below 204. We programmed the Intel Mote to indicate different link quality levels with different colors of the LED. The experiment results are shown in Table 4. The numbers in the second and third rows indicate the number of links falling in the specific link quality level for the original and the link quality–based scatternet formation schemes respectively.

Table 4 shows that the link quality in the scatternet formed with the link quality–based scatternet formation algorithm is superior to the one that tries to connect to the first tree

Table 4. Link quality distribution

Link Quality	≥ 245	$245 > LQ \geq 235$	< 204
Original	7	0.14	2.86
LQ–based	9.8	0.2	0

node responding to the inquiry regardless of the link quality (original algorithm). Most of the nodes in the scatternet formed by the link quality–based scheme has perfect or nearly perfect links connecting to the scatternet. There are 9.8 nodes connecting to the tree through links with link quality no less than 245 (0.025% BER), while no nodes connecting to the tree with high lossy links (LQ < 204). In contrast, the original algorithm forms a scatternet that has some links with higher BER than the workable limit, which results in an unreliable network.

Clearly, the network formation delay will be higher with this enhancement compared to the existing protocol. However, as mentioned earlier, in many applications the efficiency in data transfer and the reduction in link failures result in a net win in energy and delay. In our experiments, we found the scatternet formation delay for this 10 node network has increased from 113.79 to 157.81 seconds due to this enhancement.

3.4.3 Intelligent Recovery Effect

The intelligent recovery scheme for the Intel Mote is tested with a 10 node network and a 20 node network. In the experiments, all the nodes are within radio range of each other. Different nodes in the network randomly fail and the network is reconstructed. The comparison of network repair time and the number of node impacted by the failure are illustrated in Table 5.

In our experiments, we tried to cause failures in similar network topologies and at similar levels in the tree, to have more meaningful comparisons between the two approaches. It is shown in Table 5 that the intelligent repair algorithm has significant improvement in the network recovery time. For the 10 mote network, the average number of nodes affected

Table 5. Repair effect

	Original	1st level repair	nth level repair
Recovery delay(s)	82.67	11.32	34.33
No. impacted nodes	5.35	5.44	5.15

(a) 10 nodes

	Original	1st level repair	nth level repair
Recovery delay(s)	120.74	25.06	60.14
No. impacted nodes	10.89	9.6	10

(b) 20 nodes

by a node failure is about 5. The recovery time for the algorithm that disconnects all the descendant nodes is 87.67 seconds, while it is only 11.32 seconds if the first level repair is successful with the intelligent recovery algorithm. Even including the multi-level repairs, the average repair time is still much lower than the original algorithm. As the network size increases, one node failure will cause more descendant nodes to lose their connections to the root node, which incurs longer repair time. However, the repair time with the intelligent recovery scheme is still low as demonstrated in Table 5(b). With 10 nodes affected by a node failure, the repair time for the first level successful repair is only 20.76% of the repair time with the original algorithm. The larger the network size, the more significant time saving with the intelligent recovery algorithm compared to the original algorithm.

3.5 Conclusion

We have presented a metric–based scatternet formation algorithm with the consideration of the hop distance and link quality in order to improve the performance of the network constructed by the Bluetooth–based sensor devices, such as Intel Mote. We also proposed an optimization to the network repair upon node failure for a Bluetooth tree topology. Comprehensive experiments have been conducted to investigate and compare the performance of

these algorithms. The results show that with the metric–based scatternet formation scheme, the average hop distance of all the nodes in the network to the root node is greatly reduced, and the link quality in the scatternet can be guaranteed for a reliable network. The optimization for the network repair scheme has remarkable gain in the recovery delay over the original algorithm. In addition, these optimizations are easy to be incorporated into the mote without introducing excessive memory overhead and message exchange, which is important for embedded devices such as motes. It provides an example and experiences to construct wireless sensor networks with quality and reliability in consideration for the resource constrained sensor node platforms.

CHAPTER 4

SCALABILITY OF AD HOC ON–DEMAND ROUTING PROTOCOL IN VERY LARGE–SCALE MOBILE WIRELESS NETWORKS

4.1 Introduction

Mobile ad hoc networks (*MANETs*) are networks without infrastructure and having mobile nodes communicating with each other through multi–hop wireless links. In mobile ad hoc networks, devices are self–organizing, which makes its communication setup and maintenance completely different from other network solutions [59]. Each node in *MANETs* can act as a router and forward packets for others. Hence, mobile ad hoc networks can be deployed easily with a high degree of freedom and low cost.

4.1.1 Problem Description

Recent advances in wireless technologies have resulted in a large number of wireless devices participating in the ad hoc networks. Some natural civil applications, military applications, and sensor networks may involve tens of thousands of nodes. These applications may take advantages of adaptive self–organization of large–scale ad hoc networks. Scalability is a crucial property under such application environments. Different from *MANETs* operating on small scales, very large–scale *MANETs* face a number of difficulties. To detect and adjust to dynamic network conditions containing routes with nearly 100 hops or more is not an easy task.

There has been a substantial amount of research in wireless ad hoc networks, including many proposals for routing protocols [60, 61, 62, 63, 64] as well as performance evaluation and comparison of these protocols [65, 66, 67]. However, the understanding of the performance of such protocols under very large–scale ad hoc networks containing tens of thousands of nodes is relatively limited. The lack of understanding of very large–scale

ad hoc networks is primarily due to the inability of simulation tools dealing with the excessive CPU and memory requirements needed for such large networks. Therefore, most published results of *MANETs* are based on simulations of a limited number of nodes (less than 500). This may hide some potential problems of ad hoc network protocols when they are operating on large network scenarios.

4.1.2 Related Work

4.1.2.1 Scalability Study of MANETs

In the literature of mobile ad hoc networks, a major research topic is routing protocols for multi–hop ad hoc networks. A large number of routing protocols have been proposed. These protocols cover a wide range of design objectives and approaches. The efforts to evaluate and compare the performance of these ad hoc routing protocols using simulation models are presented in [65, 66, 67]. All of these works use the same *ns-2* [69] based simulation environment. Although *ns-2* is widely used, it can only comfortably support simulations with network topologies up to about 1,000 nodes with the popular routing protocols [32]. In practice, the network topologies in the performance evaluation efforts mentioned earlier are no more than 50 to 100 node models. One exceptional work [68] that studies the scalability of large scenarios of ad hoc networks using *ns-2* increases the network size up to 550 nodes. However, it may still underestimate the potential size of large–scale ad hoc networks.

A recent work evaluating the performance of mobile ad hoc network routing protocols in large–scale scenarios is presented in [70]. In this work, the *QualNet* [71] simulation environment is used to model network details. This work assesses the scalability of routing protocols by varying one control parameter at a time to stress the network in different directions. It isolates the effects of network size, network density, number of hops from sources to destinations, mobility, number of source and destination pairs, and network load on the performance of routing protocols. The largest network topology investigated was 1,000 nodes. Thus it still leaves some questions on the characteristics of ad hoc routing

protocols under very large–scale network scenarios.

To the best of our knowledge, the largest network topology for mobile ad hoc routing protocols scalability study is discussed in [72]. In that work, AODV was chosen as the protocol of choice to evaluate the scalability of routing protocols in networks as large as 10,000 nodes. The simulations are based on the models in *GloMoSim*, which is an earlier version of *QualNet*. The objective of [72] is to investigate the effects of the enhancement strategies on the performance of large–scale networks.

4.1.2.2 Wireless Network Simulators

In the research of wireless networks, simulation tools play an important role. Analyzing the performance of network protocols in very large–scale scenarios with various parameter settings is often imprecise. Furthermore, performing actual experiments with large numbers of mobile devices is very difficult. Therefore, simulators become essential for wireless network research.

The two popular simulators in the wireless network research area are *ns-2* [69] and GloMoSim [71] as we mentioned in Section 4.1.2.1. The design of *ns-2* uses OTcl as a command and configuration interface to the C++ object implementations. This approach incurs excessive memory overhead, which limits its scalability.

GloMoSim is designed for scalable wireless network simulation written in Parsec [75], a C–based simulation language. Although GloMoSim adopts the node aggregation technique, due to Parsec's large per–entity memory requirements, it has been shown to scale to 10,000 nodes [76].

Another wireless network simulator SWANS [77] is designed by researchers from Cornell University. It is built atop the JiST (Java in Simulation Time) platform. Its objective is to achieve high simulation throughput, save memory, and run standard Java network applications over simulated networks.

Our experiments are based on the *Georgia Tech Network Simulator (GTNetS)* [31, 32],

which is designed specifically for large–scale network simulations. With CPU and memory overhead in consideration, *GTNetS* can finish large–scale network simulations within reasonable time frame as we will show in Section 4.3.

4.1.3 Solution

We evaluated the potential scalability of an on–demand ad hoc network routing protocol, specifically the Ad hoc On–Demand Distance Vector (AODV) routing protocol. We chose AODV because it is a prominent on–demand routing protocol for ad hoc networks and its scalability is believed to be superior to that of other on–demand routing protocols such as Dynamic Source Routing (DSR) [61]. The poor scalability of DSR is due to the large routing overhead and stale cached routing information [68]. We ran simulation experiments using the *Georgia Tech Network Simulator (GTNetS)* [31, 32], which is a packet–level simulator designed for efficiency and scalability. We used *GTNetS* to analyze networks of up to 50,000 nodes, which is five times larger than the previous efforts. Here, we investigate the scalability of AODV with comprehensive simulation scenarios [73, 74]. All the experiments were run based on detailed models at the MAC layer, IP layer, transport layer, and application layer. Our objective was to investigate the performance of the AODV protocol in very large–scale ad hoc networks and try to exhaust the scalability limits and find the boundary of the scalability for such networks. The scalability analysis is based on various network size, node density, number of data sessions, traffic load, and mobility. Additionally, the reasons for packet loss are analyzed and categorized in detail at each network layer.

The remainder of this chapter is organized as follows. In Section 4.2, a brief description of AODV protocol is given. Section 4.3 presents the environment settings and results of our large–scale simulation experiments. Finally, the conclusions are discussed in section 4.4.

4.2 Overview of Routing Protocol

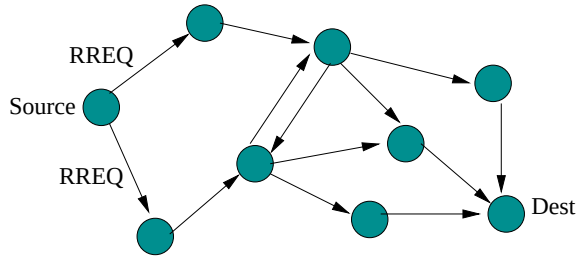
Due to the scalability superiority of AODV, we use it as a representative to evaluate routing protocols in very large–scale ad hoc networks. AODV [60] is an on–demand distance vector routing protocol. The route discovery is based on a mechanism with broadcast route requests and unicast route replies. One distinguishing feature of AODV is the use of destination sequence number for each route entry. Routing tables are maintained in each node in the network. The operation of AODV consists of route discovery, route establishment, and route maintenance activities, as shown in Figure 30. The details for the operation of the protocol are briefly discussed in the following paragraphs.

4.2.1 Route Discovery

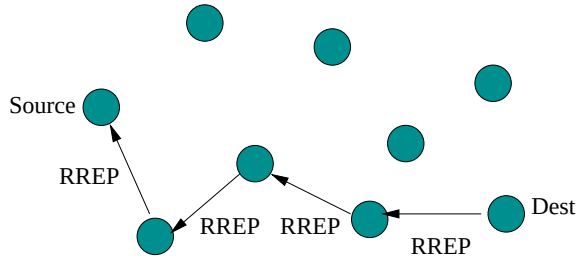
A route is needed when a source node sends a data packet to a given destination, and no route entry for the specific destination is available. In this case, a route discovery is then initiated by broadcasting route requests (RREQ). Figure 30(a) illustrates the route request flooding procedure. The RREQ includes the destination IP address, the last known sequence number of the destination, the source's IP address, and the source's sequence number. The RREQ also consists of a hop count which limits the broadcast scope, and a broadcast ID which identifies the RREQ uniquely.

When an intermediate node receives a RREQ, it creates a reverse route to the source which will be used for the route reply propagation. If the intermediate node is not the destination and has no valid route to the destination, it rebroadcasts the RREQ with an incremented hop count. In this case, the RREQ floods the network.

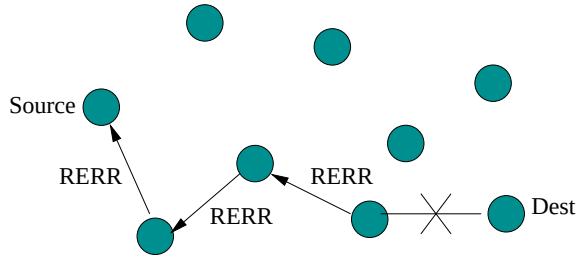
An enhancement to the RREQ flood is the expanding ring search. The intention of the expanding ring search mechanism is to find some neighbor nodes with a route to the destination in order to avoid flooding the entire network to search for a route. The operation of the expanding ring search is to set a small time to live (TTL) value to the initial RREQ, and increment for each new RREQ if no route to the destination has been found within the discovery time. This process continues until a TTL threshold is reached. After that, the



(a) RREQ broadcast



(b) RREP unicast



(c) RERR maintenance

Figure 30. AODV route discovery and maintenance

RREQ is flooded as usual. The expanding ring search introduces a tradeoff between the route discovery latency and the flooding overhead.

When the RREQ reaches the destination or an intermediate node with a valid route (with destination sequence number no less than the sequence number in the RREQ) to the destination, a route reply (RREP) is sent back to the source, which is explained in the next section.

4.2.2 Route Establishment

The destination or the intermediate node with an existing route to the destination creates a RREP which consists of the the source and destination IP address, the destination sequence number, and the lifetime of the route. The RREP is unicast back to the source using the reverse route created as the RREQ is forwarded. The RREP unicast process is indicated in Figure 30(b).

The intermediate nodes receiving the RREP create their own route entry to the destination and use the nodes from which they receive the RREP as the next hop toward the destination.

After the source receives the RREP, the route has been established and data packets can be sent to the ultimate destination.

4.2.3 Route Maintenance

Because of the characteristics of mobile ad hoc networks, an active route may break due to node mobility. In this case, a route maintenance action occurs as shown in Figure 30(c). When a link within an active route breaks, the upstream node along the route will detect the broken link and a route error (RERR) message is generated when the neighbors of the upstream node use this route. The RERR consists of a list of all the destinations that are no longer reachable because of this breakage. The neighbor nodes receiving the RERR invalidate the broken routes and generate new RERR messages based on their own neighbors using the broken link. When the source receives the RERR, it invalidates the broken routes indicated in the RERR and decides to repair it or not.

In order to detect link failures, a periodic hello message is used. A node participating in routing activities broadcasts hello messages every *Hello_Interval*. If no response is received from the neighbors, it determines that the link to the neighbor is broken and reacts by invalidating the routes using that link and generating RERR as discussed before.

The periodically transmitted hello messages consumes bandwidth of mobile ad hoc networks. Therefore, an enhancement is provided in the specification to use link layer

notifications, such as those provided by IEEE 802.11. When a packet is transmitted to the next hop, the absence of a link layer ACK or failure to receive a Clear–To–Send (CTS) after Request–To–Send (RTS) transmission is the indication of link failure. In this case, the link connectivity can be maintained without the hello message exchange.

Another enhancement of AODV is local repair of broken links. When a link failure occurs, the upstream node detecting the link failure tries to repair the route by transmitting RREQ locally. If the local repair is successful, fewer data packets will be lost and fewer control packets are transmitted for the repairing. If the local repair fails, the RERR message is generated normally.

In our experiments, we included all of these enhancements in our simulation model of AODV. In the following section, we will evaluate the performance of AODV under very large–scale ad hoc network scenarios.

4.3 Experiments

4.3.1 Simulation Model

As we mentioned in Section 4.1.2.2, the simulation model used to evaluate the performance of mobile ad hoc networks in very large–scale scenarios is implemented in *GTNetS* [31, 32]. *GTNetS* is designed specifically for modeling large–scale topologies. It has a number of features that address memory and CPU overhead which enable simulation of larger networks than is possible with other simulation tools, notably the *ns-2* simulator. The simulation modules involved in our experiments include application, transport, networking, routing, media access control, and node mobility models as shown in Figure 31. The detailed parameters are illustrated in the following sections.

4.3.1.1 General

Our simulations were performed using network sizes ranging from 10,000 mobile nodes up to 50,000 nodes. The nodes were distributed uniformly within a specified geographic region. The size of the geographic region was varied for different number of nodes in order

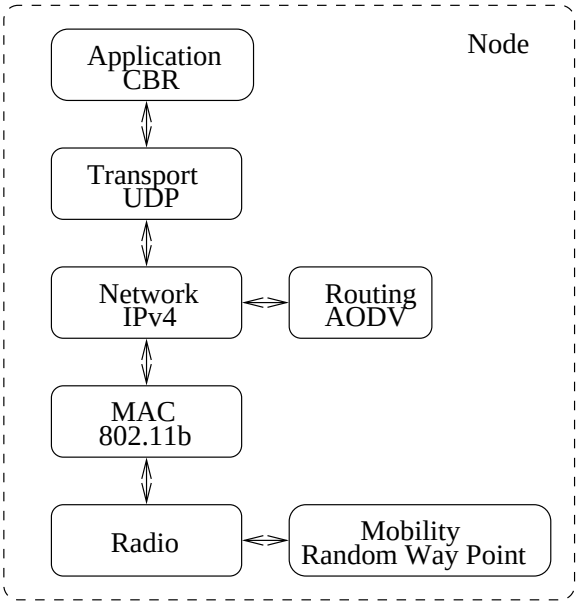


Figure 31. Simulation modules

to keep the node density approximately constant except for the experiments to investigate the effect of node density on the performance. In the node density experiments, the same number of nodes are distributed in various sizes of geographic area in order to achieve different node density.

The MAC layer protocol used in the simulations is IEEE 802.11 Distributed Coordination Function (DCF) with a channel capacity of 2Mb/s. The IEEE 802.11 model in *GTNetS* includes both RTS/CTS and virtual carrier sense features of the specification. The radio propagation range of each node was 250 meters.

Each simulation was executed for 300 seconds of simulation time. For the largest simulations with network topologies of 50,000 mobile nodes, the running time for a single simulation was about 30 hours. We ran the simulations for each scenario with different random seeds and the results are averaged over the multiple runs.

4.3.1.2 Traffic Pattern

For all the experiments in this chapter, the traffic pattern was a Constant Bit Rate (CBR) data source running on top of UDP. The packet size was 512 bytes. Twenty flows (except for the scenarios to investigate the effect of number of sources on the performance) was

simulated with randomly selected sources and destinations. For the experiments to investigate the effect of number of data sources, we varied the number of data sessions from 20 to 150. The data transmission rate was varied between 4 packets per second and 8 packets per second to investigate the effect of traffic load on the performance.

4.3.1.3 Mobility Pattern

The nodes in the simulation move according to the random waypoint model. We varied the mobility scenarios with different maximum speed and pause time. Due to long execution time of the very large–scale simulations, the maximum speed of nodes was varied between 10m/s, 20m/s, and 30m/s. The pause time was varied between 30s, 50s, and 60s.

4.3.2 Simulation Results

We present the results achieved for the different simulation scenarios in this section. The metrics used to analyze the performance of AODV in our scenarios were as follows.

- **Packet Delivery Ratio –** The packet delivery ratio is defined as the total number of data packets received at the destination divided by the number of data packets transmitted from the source.

- **End-to-End Latency –** The end–to–end latency is defined as the difference of the time stamp when a data packet leaves a source node and the time stamp as it arrives at the destination. It is averaged over all successfully received data packets.

- **Control Overhead –** The control overhead in AODV includes route request (RREQ), route reply (RREP), and route error (RERR) messages. Each hop–wise transmission of a control message by a node is considered as one control packet.

- **Average Hop Count –** The average hop count is defined as the number of hops along a path averaged on all the successfully established routes.

In addition, for each cause of packet loss, we measured their effects on the performance individually. The simulation results are illustrated in the following sections.

4.3.2.1 Packet Delivery Ratio

This set of experiments are designed to evaluate the fraction of successful packet delivery under different traffic loads and mobility scenarios with various number of nodes. The results are shown in Figure 32. In the figures hereafter, we denote the normal simulation scenario as the case with data transmission rate of 4 packets/s, maximum speed 10 m/s, and pause time 30 seconds. The other scenarios are indicated with the parameters varied.

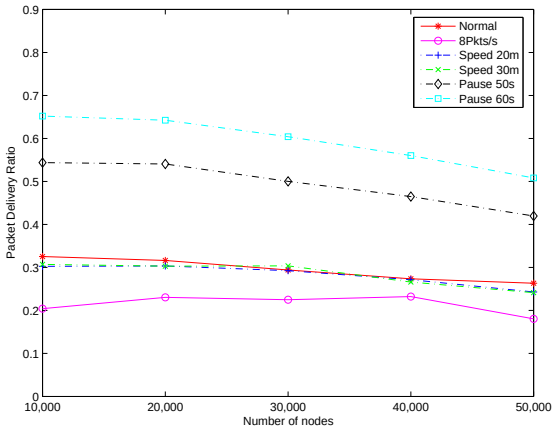


Figure 32. Packet delivery ratio

Figure 32 shows that the variation in the packet delivery ratio as a function of topology size is small. This is because it is already very hard to establish and maintain routes in networks with 10,000 nodes. With the normal simulation setting, the packet delivery ratio for 10,000 nodes networks is only 32.5%. The packet delivery ratio in 50,000 nodes networks for the same setting is only 26.3%.

The effect of traffic load on the packet delivery ratio is noticeable however. With the traffic generation rate of 8 packets/s, the packet delivery ratio degrades 31.5% compared to the normal scenario with packet generation rate of 4 packets/s for networks with 50,000 nodes. The reason that high network load plays a significant part on the packet delivery ratio is that more packets are transmitted in the network and competing for access to the channel, which increases the likelihood of collisions.

Mobility is an important issue affecting the performance and scalability of ad hoc networks. Frequent link breakage and route recovery due to node mobility limit the scalability of mobile ad hoc networks. To investigate the effect of node mobility on the packet delivery ratio, we varied both the maximum speed and the pause time. We ran the simulations with maximum speeds of 10m/s, 20m/s, and 30m/s. The packet delivery ratio shown in Figure 32 dictates that the effect of the maximum speed on the packet delivery ratio is not significant. On the contrary, the pause time has a substantial impact on the packet delivery ratio. For higher pause time (low mobility) of 60 seconds, the packet delivery ratio is 50.8% for the network topologies with 50,000 nodes, while it is 42.0% for pause time of 50 seconds and only 26.3% for pause time of 30 seconds. Therefore, as long as the nodes keep moving in such large networks, the chances of link breakage on the long route are high regardless of the speed. The packet delivery ratio only increases when the nodes pause for a long time.

Based on the above analysis, the packet delivery ratio for the very large–scale mobile ad hoc networks with node from 10,000 to 50,000 is very close for the same parameter settings. The traffic load and the pause time have significant effect on the performance whereas the effect of the maximum speed does not.

4.3.2.2 End–to–End Latency

Figure 33 shows the results of the end–to–end latency of each simulation scenario.

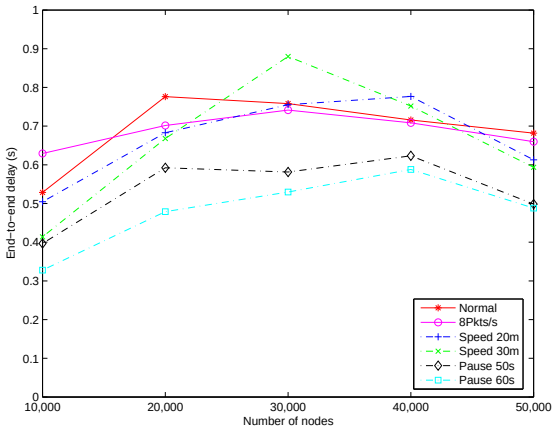


Figure 33. End–to–end latency

It can be observed that the end–to–end delay for all the simulation cases increases sharply when the number of nodes in the networks is increases from 10,000 to 20,000. When the network size grows beyond 20,000 nodes, the end-to-end delay is fluctuated with small variances for different simulation scenarios. The contributions to the end-to-end latency are the buffering delay experienced by packets waiting for the discovery of a new route, transmission delay of longer route with large number of hops, and the route recovery delay. From Figure 33, it can be observed that the degree of mobility in the term of pause time has a larger impact on the performance of end–to–end delay than any other factors. As an example, the average end-to-end delay for a pause time of 60 seconds is 479ms, whereas it is 776ms for a pause time of 30 seconds with the network size of 20,000 nodes. The effects of traffic load and the maximum speed on the end–to–end latency is not obvious. This is the outcome with the combination of the packet delivery ratio and the total packets arriving at the destinations.

4.3.2.3 Control Overhead

One major obstacle for the scalability of mobile ad hoc network routing protocols is too much control overhead due to the very large topologies, high mobility, and frequent route maintenance. Therefore, the control overhead is an important metric to measure the performance of mobile ad hoc routing protocols.

In our simulations, we gathered statistics of the control overhead including route request (RREQ), route reply (RREP), and route error (RERR) messages. The results of the control overhead are demonstrated in Figure 34.

The control overhead measured as the number of control packets sent per data packet received increases as the network size expands. For the network size less than 40,000 mobile nodes, the increase of the control overhead is almost linear. When the network size reaches 50,000 mobile nodes, the control overhead rises steeply. In addition, the differences of control overhead under various simulation scenarios are significant for the network size with 50,000 nodes, whereas they are small for the network size with less than 40,000 nodes.

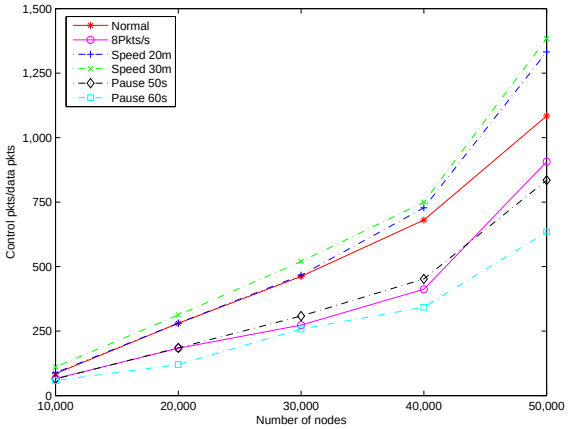


Figure 34. Overhead of control packets

The impact of the traffic load and the mobility on the control overhead can be seen from Figure 34. Similar to the trend of the packet delivery ratio discussed in section 4.3.2.1, the variances of the control overhead for various maximum speeds are not significant. On the contrary, with lower mobility (longer pause time) or higher traffic load, the control overhead is noticeably lower than the normal scenario. This is because lower mobility means fewer link failures, hence fewer route repair messages. When the traffic load is high, more data packets are generated from the same source and have the same destination. In this case, more data packets can share the results of a route discovery and fewer RREQ and RREP are required.

4.3.2.4 Hop Distance

In very large–scale mobile ad hoc networks, the route length from the sources to the destinations can be excessive. It is difficult to maintain such long routes in mobile environments. In addition, the large hop distance also contributes to the end–to–end latency. We measured the average hop distance for the networks with 10,000 to 50,000 nodes and presented the results in Figure 35.

The performance of average hop distance shown in Figure 35 dictates that the hop distance increases steeply from 58.5 hops for the network size of 10,000 nodes to 88.5

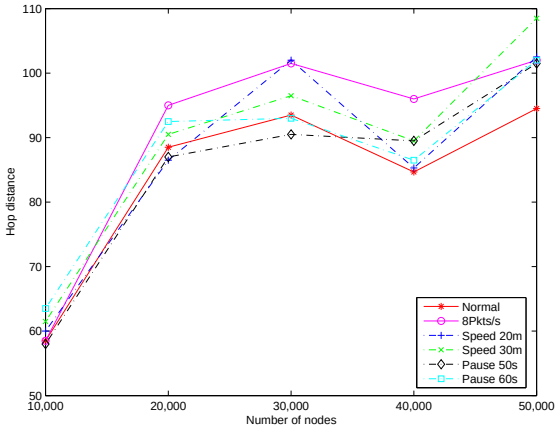


Figure 35. Hop distance

hops for the network size of 20,000 nodes. With the network size larger than 20,000 nodes, the average hop distance does not vary too much. This trend is coincident with the end–to–end delay we discussed in section 4.3.2.2. This is not by chance. It is because the long route with large hop distance means huge transmission delay and high possibility of route failure, leading therefore to large end–to–end latency.

4.3.2.5 Node Density

The density of nodes in a network has an effect on the performance of ad hoc routing protocols. These experiments analyze this effect by increasing the node density while keeping the number of nodes and the number of sources fixed and varying the distance between neighboring nodes in the network. Due to the excessively long running time, we only simulate networks with 10,000 nodes. The results of packet delivery ratio (PDR), end–to–end delay, and average hop count are shown in Table 6.

Table 6. Effect of node density (10,000 nodes)

Density (# of neighbors)	7	10	15	20
PDR(%)	32.54	29.03	31.58	31.50
Delay(s)	0.529	0.422	0.365	0.330
Hops	58.5	42.67	40	32.67

When the node density is increased from 7 to 20 neighbors per node, the packet delivery ratio stays approximately the same. This is because for the networks with tens of thousands of nodes, the density of nodes is not the major contributor to the packet delivery ratio as long as there are enough neighbors to keep the network connected. On the other hand, the end–to–end delay is decreased with increased node density. The reason is that with high node density, there are more candidates along the path for route discovery and AODV always prefers the route with the shortest delay. So the end–to–end latency is reduced. The average hop distance from a source to a destination becomes small when the node density increases. This is also the result of multiple paths available with greater node density. In AODV, the selected route with the lowest delay corresponds to the route with fewer hop distance most of the time. As a conclusion, for very large–scale ad hoc networks, the node density has little impact on the packet delivery ratio. However, the end–to–end delay and average hop count can be improved in dense networks.

4.3.2.6 Number of Sources

The number of sources and destinations in the network has an effect on the performance of the routing protocol. The larger number of sources in the network, the more routes are required to be maintained. In the previous experiments, the number of data source and destination pairs is kept constant to limit the number of variables. To investigate the effect of number of sources, we varied the number of data sessions in the network. The simulation results with 10,000 nodes are illustrated in Table 7.

Table 7. Effect of number of sources (10,000 nodes)

Num of src	20	50	100	150
PDR(%)	32.54	22.20	18.15	15.78
Delay(s)	0.528	0.528	0.658	0.707
Hops	58.5	47.4	49.2	49.6

It can be observed from Table 7 that the network performance in the term of packet delivery ratio degrades as the number of data sessions in the network increases. The more

data sessions in the network, the higher probability of packet collisions, hence the lower packet delivery ratio. Because of the increased chance of packet collision, it also takes longer time for route discovery. Therefore, the average end–to–end delay grows with the number of data sessions within the network. Unlike the network density, the number of data sources has less impact on the average hop distance.

4.3.2.7 Packet Loss Categories

In the above sections, we evaluated the performance of mobile ad hoc routing protocols in very large–scale networks from various perspectives including packet delivery ratio, end–to–end latency, control overhead, and hop distance under different network scenarios. The factor affecting the performance is the packet loss by origin. In this section, we will discuss the causes for packet losses and break them into detail categories in separate network layers.

In the simulations, we considered each of the reasons for packet losses at different network layers and tried to optimize the parameter settings to reduce the packet losses. The optimizations includes the size of the queue for the packets to be buffered when they are awaiting for the route discoveries, and the interface queue size for the packets to be buffered at MAC layer. After these optimizations, the major causes for the packet losses can be categorized as follows:

- **AODV Queue Timeout –** In AODV, data packets waiting for a route RREP after sending a RREQ should be buffered. The is a first–in first–out (FIFO) queue. After a route discovery attempt is successful, all the packets using the discovered route are dequeued and sent to the destination.

  When the data packets are put into the queue and the queue is full, the first packet in the queue with expired time stamp will be deleted from the queue. In the simulations, the timeout value for the data packets residing in the AODV queue is 30 seconds. The packet drop because of expiration in the AODV queue is categorized as AODV queue timeout.

- **RREQ Failure –** The route discovery in AODV may not be successful in the first attempt. The maximum number of route discovery retries (*RREQ_RETRIES*) is specified in AODV with the default setting of 3 times. For every retry, the waiting time for the RREP is binary exponentially increased to reduce congestion in a network. If the route discovery has been attempted for *RREQ_RETRIES* times without receiving any RREP, all the data packets destined for the corresponding destination should be dropped from the buffer. This is the RREQ failure drop.

- **Network Layer Drop –** A data packet dropped in the IP layer is due to node mobility. In very large–scale networks with tens of thousands of mobile nodes, the average hop distance of the routes from the sources to the destinations can be one hundred or more. To maintain the connectivity of such a long route is very difficult. Even with AODV local repair, when several nodes along a route move out of the radio range of their neighbors, it is hard for the local repair to be synchronized and the route to be recovered. Therefore, the network layer drop in the very large–scale mobile ad hoc networks is significant.

- **MAC Layer Drop –** The MAC layer protocol used in the simulations is IEEE 802.11 Distributed Coordination Function (DCF). The channel access scheme is Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) with acknowledgments. Before unicast data transmissions, Request–To–Send(RTS)/Clear–To–Send(CTS) messages are exchanged for channel reservation. After successful data transmissions, ACKs are sent for acknowledgments. The MAC layer drops occur due to contention and a neighbor moving out of range, leading to an RTS/CTS failure or unacknowledged data packets.

In our simulations, we kept statistics on the packet drops due to the categorized reasons discussed above. We compared the packet drop of networks with 50,000 nodes under various simulation scenarios designed for traffic load and mobility investigation. Figure

36 shows the simulation results.

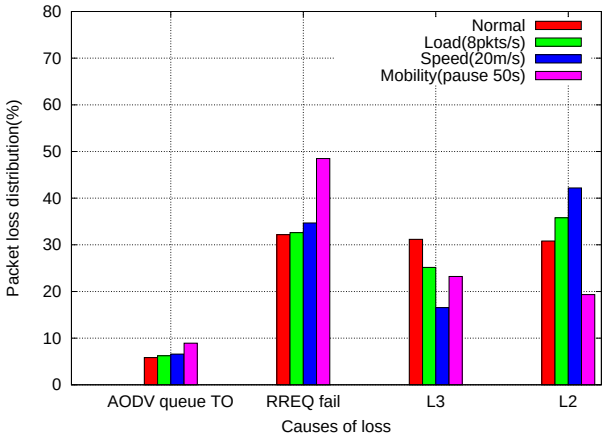


Figure 36. 50K nodes packet loss categories

It can be observed from Figure 36 that in all of the four simulation scenarios with various traffic loads and mobilities, the packet loss due to packet timeout in AODV queue awaiting for the results of route discovery is relatively low (less than 10%). The other three reasons play more important roles resulting in packet drops.

For the scenario with normal parameter setting, the RREQ failure, network layer drop, and MAC layer drop have almost the same effects (around 30%) on the packet losses. When the traffic load is increased with packet generating rate of 8 packets/s, the MAC layer drop is more significant than the other two. This is because higher traffic load means more collisions at the MAC layer, hence more packet losses. On the contrary, the percentage of packet loss at network layer for higher traffic load reduces due to more data packets sharing the same route discovery.

The effect of node mobility on the packet drop metrics is illustrated with the scenarios of increased maximum speed (20m/s) and large pause time (50 seconds) respectively. When the maximum speed increments to 20m/s, the MAC layer packet loss plays a dominant role (42.4%) whereas the network layer packet drop decreases. Therefore, the packet drop in the MAC layer is very sensitive to the mobility speed of the nodes in the networks. The effect of the other mobility parameter, pause time, is different from the speed. With longer

pause time, the majority of packet drops are due to RREQ failure, which is 48.5%. The reason behind this is that there are fewer link breaks with less mobility. So the packet drops in both the network layer and the MAC layer are not significant.

In order to distinguish the effect of different causes of packet drop on the networks of various sizes, we compared the packet drop categories among networks with size of 10,000 nodes to 50,000 nodes. The result is presented in Figure 37.

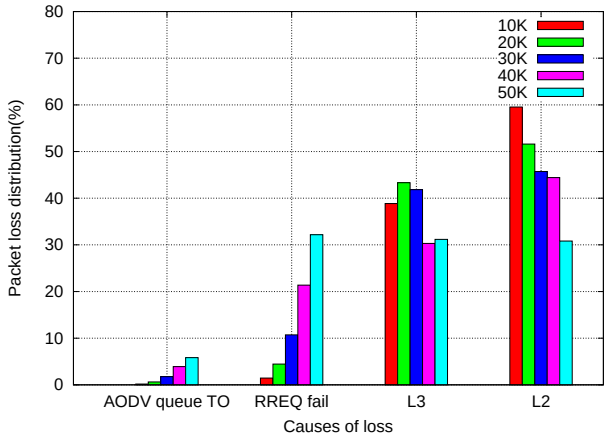


Figure 37. Packet loss comparison

Figure 37 shows that as the network size increases, the effects of AODV queue timeout and RREQ failure become more and more severe whereas the effects of the network layer drop and the MAC layer drop reduce. This is because for the smaller networks, the average hop distances of the routes is relatively short compared to the networks with 50,000 nodes. So the route discovery time and the possibility of RREQ failure is decreased. For the networks with 40,000 or 50,000 nodes, the same number of data flows spread within larger networks. Therefore, there are fewer collisions in these cases, which results in less MAC layer packet drops.

4.4 Conclusions

Scalability of the routing protocol in very large–scale mobile ad hoc networks is difficult due to the excessively long route and node mobility. We investigated in detail the scalability of on–demand routing protocols, using AODV as a representative sample, for very

large–scale mobile ad hoc networks with up to 50,000 mobile nodes. We designed a set of comprehensive simulations to address the scalability analysis from different aspects, which include the effect of the network size, node density, number of data sessions, traffic load, and mobility. This unprecedented work was conducted using our *GTNetS* simulation environment, which makes it possible to simulate and study the performance of such large–scale networks in a reasonable time period.

We presented the scalability performance of AODV from the aspects of packet delivery ratio, end–to–end latency, control overhead, and hop distance. A detailed analysis of the simulation results is presented from both quantitative and qualitative perspectives. In particular, in order to understand the fundamental reasons affecting the performance of these large–scale ad hoc networks, we isolated the causes for packet drops at various network layers and measured them for different network scenarios. This data can provide a guideline for the protocol design and future enhancements.

We provided a methodology to investigate the performance and scalability bounds of mobile ad hoc networks. However, scalability in very large–scale mobile ad hoc networks still leaves many uncovered areas. More scalable and light weight routing protocols for very large–scale mobile ad hoc networks are necessary.

CHAPTER 5

CONCLUSIONS

The objective of this research is to design a self–organized, power–efficient, low–delay, and reliable Bluetooth scatternet formation and routing protocol for the high data rate, low duty cycle wireless sensor network applications, as well as to investigate the scalability of ad hoc routing protocols. More specifically, an on–demand scheme combining scatternet formation and routing is designed with the following contributions: a modified *Inquiry* process with extended ID packets is proposed for Bluetooth symmetric device discovery and efficient route request propagation; a state switch scheme alternating between *Page_Scan* and *Inquiry_Scan* is exploited to support multiple source initiating traffic and scatternet formation simultaneously; the modified *POLL* packets in *Page* process are introduced for scatternet formation and route reply information transfer without extra expenses; an energy–aware forwarding node selection scheme is used to prolong the network lifetime. In addition, several optimizations including the metric–based scatternet formation scheme and an intelligent recovery mechanism are proposed for Bluetooth–based sensor motes such as Intel Mote. Furthermore, the scalability study of ad hoc routing protocol in very large–scale scenarios is conducted.

5.1 Summary of Research Results

5.1.1 Bluetooth Simulator

A simulation environment for wireless sensor networks using the Bluetooth wireless protocol in the *Georgia Tech Network Simulator (GTNetS)* has been developed. The Bluetooth simulator implements detailed behavior of the lower layers of the Bluetooth protocol stack, including Baseband, LMP, L2CAP, and BNEP, with an emphasis on interference resilient and power efficient characteristics. It provides a simulation environment to investigate the performance of wireless sensor network applications running on top of Bluetooth. Some

experiment results in [30] with the Bluetooth simulator show the power efficiency of Bluetooth's low power modes, which makes it applicable for high data rate and low duty cycle wireless sensor network applications. The power measurement level provides a guideline for protocol improvement and sensor node lifetime estimation.

5.1.2 Energy–Aware On–Demand Scatternet Formation and Routing Protocol

An on–demand scatternet formation and routing protocol used for Bluetooth–based wireless sensor network applications has been proposed. It deals with the complicated problem of supporting multiple sources that initiate the scatternet and route formation involving common Bluetooth devices at the same time. The scatternet formation and routing protocol includes several aspects:

5.1.2.1 Modified Inquiry with EID packets

A modified Bluetooth *Inquiry* process with extended ID (EID) packets for scatternet formation and route request propagation was introduced. It solves the problem of Bluetooth asymmetric device discovery and accelerates the process as evaluated in [43]. The modified *Inquiry* is employed to propagate the source and last hop information in the downstream direction from the source to the sink during scatternet formation and route discovery phase. The EID packet is used in the modified *Inquiry* rather than replacing the original ID packet to save energy, which is important for wireless sensor network applications. Simulation results [43, 44] showed the power efficiency and promptness of the modified *Inquiry* scheme compared with traditional *Inquiry*.

5.1.2.2 State Switch for Multiple Simultaneous Sources

A state switch scheme alternating between *Page_Scan* and *Inquiry_Scan* for the nodes that have participated in the route discovery in the scatternet formation process was proposed to enable multiple sources to start route requests simultaneously. It allows the intermediate nodes to wait for the scatternet and route formation replies in *Page* messages from the downstream nodes. At the same time, the nodes in route discovery can still accept new

route requests from other sources. The performance in terms of scatternet formation delay achieves significant improvement when compared to serial scatternet formation for multiple sources with concurrent traffic, which is shown in [44, 45].

5.1.2.3 Modified Page with POLL packets

A mechanism using *POLL* packets in *Page* mode to transfer scatternet formation information without extra expenses was proposed. In the on–demand scatternet formation approach, a strict master/slave role alternation is maintained along any route from a source to the sink. Therefore, in the scatternet connection establishing phase, when a node with a *ScatRole* of slave tries to page its last hop nodes to relay the scatternet formation information, the *AMAddr* in the *POLL* packet is set to zero, while the *AMAddr* assigned by nodes with *ScatRole* of master is between one and seven. In this case, the upstream nodes in the scatternet route can decide their *ScatRoles* based on the *AMAddr*. With the modified *Page* scheme, *ScatRole* information is transfered without overhead and there is no effect on the proper operation of the scatternet formation.

5.1.2.4 Energy–Aware Forwarding Node Selection

An energy–aware forwarding node selection scheme based on local information was proposed to balance the traffic load over the entire network and prolong the network lifetime. The selection of relaying nodes from the sensor sources to the sink makes use of the information of residual current in the intermediate nodes and tries to make the energy consumption of all the nodes within the network approximately the same and to avoid extra load at any given nodes. Simulation results [44, 45] demonstrate that the load balance scheme extends the network lifetime. It meets the requirements of Bluetooth–based wireless sensor networks in terms of power efficiency and uniform resource utilization.

5.1.3 Metric–Based Scatternet Formation and Recovery Optimization for Intel Mote

A metric–based scatternet formation algorithm is proposed with the consideration of the hop distance and link quality in order to improve the performance of the network constructed by the Bluetooth–based sensor motes, such as Intel Mote. In addition, an optimization to the network repair upon node failure is presented for efficient network maintenance.

5.1.3.1 Hop Distance–Based Scatternet Formation

A hop distance–based scatternet formation scheme is proposed for the Bluetooth–based Intel Mote. Because of the connection–oriented characteristics of Bluetooth, the increase in hop count in a Bluetooth scatternet results in more piconet switches along the route, which further increases the forwarding traffic and delay. The hop distance–base scatternet formation strictly limits the hop distance to the root node in the beginning of the scatternet formation phase and looses this restriction as the process goes on. The hop distance–based scatternet formation algorithm improves the performance of the constructed scatternet by trading some scatternet formation delay. For many industrial applications that the formed topology is maintained for a long time, the extra delay for the scatternet formation is negligible and more than compensated.

5.1.3.2 Link Quality–Based Scatternet Formation

A link quality–based scatternet formation algorithm is proposed for the Bluetooth–based Intel Mote. Similar to the hop distance–based scatternet formation scheme, this algorithm tries to look for the best quality link available to construct the network. The link quality–based scatternet formation scheme does not require local storage of link quality information of the neighbors. It improves the reliability of data communication in the network and reduces network disconnections.

5.1.3.3 Intelligent Recovery

An intelligent network recovery scheme is proposed for the Bluetooth tree topology. Instead of disconnecting all the descendant nodes in a subtree upon link or node failure, we

maintain the subtree architecture by putting the descendant nodes into Bluetooth low power mode hold, while the network repair is conducted by the node detecting the failure. Once the repair is successful, all the connections of the descendant nodes are resumed by transmitting a routing refresh message. Otherwise, the nodes in the next level of the tree will initiate the repair. With our enhanced recovery algorithm, the network recovery time from a node or link failure is greatly reduced compared to the algorithm that disconnect all the descendant nodes.

5.1.4 Scalability Study of On–Demand Ad Hoc Routing Protocol

To investigate the potential scalability of on–demand ad hoc routing protocol, specifically the Ad hoc On–Demand Distance Vector (AODV) routing protocol, comprehensive experiments are conducted in networks with up to 50,000 nodes. The scalability analysis is based on various network size, node density, number of data sessions, traffic load, and mobility. In order to understand the fundamental reasons affecting the performance of these very large–scale ad hoc networks, the reasons for packet loss are analyzed and categorized in detail at each network layer. This study can provide a guideline for the protocol design and enhancement for very large–scale ad hoc routing protocols.

REFERENCES

[1] IEEE Computer Society, "IEEE standard 802.15.4-2003," Available HTTP: http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf, Oct. 2003.

[2] Bluetooth SIG, "Bluetooth specification version 1.1," Available HTTP: http://www.bluetooth.com, Nov. 2003.

[3] E. Ferro, F. Potorti, "Bluetooth and Wi-Fi wireless protocols: a survey and a comparison," in *IEEE Wireless Communications,* vol. 12, no. 1, pp. 12-26, 2005.

[4] T.M. Siep, I.C. Gifford, R.C. Braley, and R.F. Heile, "Paving the way for personal area network standards: an overview of the IEEE P802.15 working group for wireless personal area networks," in *IEEE Personal Communications,* vol. 7, no. 1, pp. 37-43, 2000.

[5] F. Zhao and L. Guibas, *Wireless sensor networks: an information processing approach.* San Francisco, CA: Morgan Kaufmann Publishers, 2004.

[6] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," in *Computer Networks,* vol. 38, no. 4, pp. 393-422, 2002.

[7] I.F. Akyildiz, I.H. Kasimoglu, "Wireless sensor and actor networks: research challenges," in *Ad Hoc Networks,* vol. 2, pp. 351-367, 2004.

[8] Intel "Health Research and Innovation," Available HTTP: http://www.intel.com/research/prohealth, Mar. 2006.

[9] IBM "m-Health Wireless," Available HTTP: http://www-03.ibm.com/technology/designconsulting/port_mhealth.html, Mar. 2006.

[10] IEEE Computer Society, "IEEE standard 802.11-1999," Available HTTP: http://standards.ieee.org/getieee802/download/802.11-1999.pdf, Mar. 1999.

[11] M. Leopold, M.B. Dydensborg, and P. Bonnet, "Bluetooth and sensor networks: a reality check," in *Proc. 1st International Conference on Embedded Networked Sensor Systems,* 2003, pp. 103-113.

[12] P. Bonnet, A. Beaufour, M.B. Dydensborg, and M. Leopold, "Bluetooth-based sensor networks," *ACM SIGMOD Record, Special Issue: Special Section on Sensor Network Technology and Sensor Data Management,* vol. 32, no. 4, pp. 35-40, 2003.

[13] Intel, "Sensor nets/RFID," Available HTTP: http://www.intel.com/research/exploratory/motes.htm, Mar. 2006.

[14] J. Beutel, O. Kasten, F. Mattern, K. Romer, F. Siegemund, and L. Thiele, "Prototyping wireless sensor network applications with BTnodes," in *Proc. 1st IEEE European Workshop on Wireless Sensor Networks (EWSN),* 2004, pp. 323-338.

[15] R.M. Kling, "Intel Motes: an enhanced sensor network node," in *Proc. International Workshop on advanced sensors, structural health monitoring, and smart structures,* 2003.

[16] R.M. Kling, R. Adler, J. Huang, V. Hummel, and L. Nachman, "Intel Motes: using Bluetooth in sensor networks," in *Proc. 2nd International Conference on Embedded Networked Sensor Systems (SenSys),* 2004, pp. 318-318.

[17] R.M. Kling, R. Adler, J. Huang, V. Hummel, and L. Nachman, "Intel Mote: sensor network technology for industrial applications," in *Proc. 24th IEEE INFOCOM,* 2005.

[18] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA),* 2002, pp. 88-97.

[19] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Proc. 1st European Workshop on Wireless Sensor Networks (EWSN),* 2004, pp. 307-322.

[20] V. Mehta, M.E. Zarki, "A Bluetooth based sensor network for civil infrastructure health monitoring," in *ACM/Kluwer Journal on Wireless Networks (WINET),* vol. 10, no. 4, pp. 401-412, 2004.

[21] M. Handy, J. Blumenthal, and D. Timmermann, "Energy-efficient data collection for Bluetooth-based sensor networks," in *Proc. IEEE Instrumentation and Measurement Technology Conference (IMTC),* 2004.

[22] G. Zaruba, S. Basagni, and I. Chlamtac, "Bluetree - scatternet formation to enable Bluetooth-based personal area networks," in *Proc. IEEE International Conference on Communications (ICC),* 2001, pp. 273-277.

[23] Z. Wang, R.J. Thomas, and Z. Haas, "BlueNet - a new scatternet formation scheme," in *Proc. 35th Hawaii International Conference on System Science (HICSS),* 2002.

[24] C. Petrioli, S. Basagni, and I. Chlamtac, "Configuring BlueStars: multihop scatternet formation for Bluetooth networks," in *IEEE Transaction on Computers: Special Issue on Wireless Internet,* vol. 52, no. 6, pp. 779-790, 2003.

[25] C. Petrioli, S. Basagni, and I. Chlamtac, "Bluemesh - degree-constrained multihop scatternet formation for Bluetooth networks," in *ACM/Kluwer Journal on Mobile Networks and Applications (MONET),* vol. 9, no. 1, pp. 33-47, 2004.

[26] S. Basagni, R. Rbuno, G. Mambrini, and C. Petrioli, "Comparative performance evaluation of scatternet formation protocols for networks of Bluetooth devices," in *ACM/Kluwer Journal on Wireless Networks (WINET),* vol. 10, no. 2, pp. 197-213, 2004.

[27] Y. Liu, M.J. Lee, and T.N. Saadawi, "A Bluetooth scatternet-route structure for multihop ad hoc networks," *IEEE JSAC,* vol. 21, no. 2, pp. 229-239, 2003.

[28] E. Pagani, G.P. Rossi, and S. Tebaldi, "An on-demand Bluetooth scatternet formation algorithm," in *Proc. 1st IFIP Working Conference on Wireless On-demand Network Systems (WONS),* 2004, pp. 130-143.

[29] Y. Kawamoto, V. Wong, and V. Leung, "A two-phase scatternet formation protocol for Bluetooth wireless personal area networks," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC),* 2003, pp. 1453-1458.

[30] X. Zhang and G.F. Riley, "Bluetooth simulations for wireless sensor networks using GTNetS," in *Proc. 12th IEEE International Symposium on Modeling, Analysis, and Simulation of i Computer and Telecommunication Systems (MASCOTS),* 2004, pp. 375-382.

[31] G.F. Riley, "The Georgia Tech network simulator," in *Proc. ACM SIGCOMM Workshop on Models, Methods, and Tools for Reproducible Network Research,* 2003, pp. 5-12.

[32] G.F. Riley, "Large-scale network simulations with GTNetS," in *Proc. 2003 Winter Simulation Conference,* 2003, pp. 676-684.

[33] Georgia Tech, "GTNetS," Available HTTP: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/, 2006.

[34] T. salonidis, P. Bhagwat, L. Tassiulas, and R. Lamaire, "Distributed topology construction of Bluetooth personal area networks," in *Proc. 20th IEEE INFOCOM,* 2001, pp. 1577-1586.

[35] S. Basagni, R. Bruno, and C. Petrioli, "Performance evaluation of a new scatternet formation protocol for multi-hop Bluetooth networks," in *Proc. 5th International Symposium on Personal Wireless Multimedia Communications (WPMC),* 2002, pp. 208-212.

[36] F. Xue and P.R. Kumar, "The number of neighbors needed for connectivity of wireless networks," in *ACM/Kluwer Wireless Networks (WINET),* vol. 10, no. 2, pp. 169-181, 2004.

[37] F. Ferraguto, G. Mambrini, A. Panconesi, and C. Petrioli, "A new approach to device discovery and scatternet formation in Bluetooth networks," in *Proc. 18th International Parallel and Distributed Processing Symposium (IPDPS),* 2004, pp. 221-228.

[38] T. salonidis, P. Bhagwat, and L. Tassiulas, "Proximity awareness and fast connection establishment in Bluetooth," in *Proc. 1st ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc),* 2000, pp. 141-142.

[39] T. salonidis, P. Bhagwat, L. Tassiulas, and R. Lamaire, "Proximity awareness and ad hoc networks establishment in Bluetooth," in Technical Report, Institute of Systems Research (ISR), University of Maryland, Cambridge, MA, vol. TR 2001-10, 2001.

[40] D.M. Blough, P. Santi, "Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks," in *Proc. 8th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM),* 2002, pp. 183-192.

[41] Crossbow Technology, "MICA2 AA battery pack service life test," Available HTTP: http://www.crossbow.com/Support, May 2004.

[42] J. Linsky, "Bluetooth and power consumption: issues and answers," Available HTTP: http://www.rfdesign.com, 2001.

[43] X. Zhang, G.F. Riley, "Evaluation and accelerating Bluetooth device discovery," in *Proc. IEEE Radio and Wireless Symposium (RWS),* Jan. 2006.

[44] X. Zhang, G.F. Riley, "An on-demand Bluetooth scatternet formation and routing protocol for wireless sensor networks," in *Proc. 1st ACIS International Workshop on Self-Assembling Wireless Networks (SAWN),* May 2005.

[45] X. Zhang, G.F. Riley, "Energy aware on-demand scatternet formation for Bluetooth-based wireless sensor networks," in *IEEE Communications Magazine, Adhoc and Sensor Networks Series,* vol. 43, no. 7, pp. 126-133, July 2005.

[46] L. Nachman, R.M. Kling, R. Adler, J. Huang, and V. Hummel, "The Intel Mote platform: a bluetooth-based sensor network for industrial monitoring," in *Proc. International Conference on Information Processing in Sensor Networks (IPSN),* Apr. 2005, pp. 437-442.

[47] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," in *ACM Communications,* vol. 47, no. 6, pp. 41-46, 2004.

[48] J. Hill, D. Culler, "System architecture for wireless sensor networks," *Ph.D thesis, University of California, Berkeley,* May 2004.

[49] J. Hill, D. Culler, "Mica: a wireless platform for deeply embedded networks," in *IEEE Micro,"* vol. 22, no. 6, pp. 12-24, 2002.

[50] J.L. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in Proc. 9th International Conference on Architectural Support Programming Languages and Operating Systems (ASPLOS-IX), Nov. 2000, pp. 93-104.

[51] Intel "Stargate," Available HTTP: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0049-01_B_STARGATE.pdf, Feb. 2006.

[52] R. Adler, P. Buonadonna, et al., "Design and deployment of industrial sensor networks: experiences from the north sea and a semiconductor plant," in *Proc. 3rd ACM Conference on Embedded Networked Sensor systems (SenSys),* Nov. 2005.

[53] C.K. Kallo, M. Brunato, S. Jung, L.J. Chen, and M. Gerla, "Throughput, energy and path length tradeoffs in Bluetooth scatternets," in *Proc. IEEE International Conference on Communications (ICC),* May 2005.

[54] J. Beutel, "Robust topology formation using BTnodes," in *Computer Communications,* vol. 28, no. 13, pp. 1523-1530, Aug. 2005.

[55] G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, "An efficient scatternet formation algorithm for dynamic environments," in *Proc. IASTED Communications and Computer Networks (CCN),* Nov. 2002.

[56] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proc. 1st International Conference on Embedded Networked Sensor systems (SenSys),* Nov. 2003, pp 14-27.

[57] A. Madhavapeddy and A. Tse, "A study of Bluetooth propagation using accurate indoor location mapping," in *Proc. 7th International Conference on Ubiquitous Computing (Ubicomp),* Sep. 2005.

[58] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proc. ACM Conference on Information Processing in Sensor Networks (IPSN),* Apr. 2005, pp. 364-369.

[59] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, *Mobile ad hoc networking.* Piscataway, NJ: IEEE Press, Wiley Interscience, Inc., 2004.

[60] C.E. Perkins, E.M. Belding-Royer, and S. Das, "Ad hoc on demand distance vector (AODV) routing," *IETF RFC 3561,* July 2003.

[61] D.B. Johnson, D.A. Maltz, and Y.C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)," *IETF Draft, draft-ieft-manet-dsr-09.txt,* Apr. 2003.

[62] C.E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," in *Proc. ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications,* 1994, pp. 234-244.

[63] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," *IETF RFC 3626,* Oct. 2003.

[64] Z.J. Haas and M.R. Pearlman, "The zone routing protocol (ZRP) for ad hoc networks," *Internet Draft, draft-ietf-manet-zone-zrp-02.txt,* June 1999.

[65] C.E. Perkins, E.M. Belding-Royer, and S. Das, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal Communications,* vol. 8, no. 1, pp. 16-28, 2001.

[66] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc routing protocols," in *Proc. 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM),* 1998, pp. 85-97.

[67] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks," in *Proc. 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM),* 1999, pp. 195-206.

[68] V. Naumov and T. Gross, "Scalability of routing methods in ad hoc network," *International Journal of Performance Evaluation,* vol. 62, pp. 193-209, 2005.

[69] ISI, "The network simulator - ns-2," Available HTTP: http://www.isi.edu/nsnam/ns/, 2003.

[70] J. Hsu, S. Bhatia, K. Tang, R. Bagrodia, and M.J. Acriche, "Performance of mobile ad hoc networking routing protocols in large scale scenarios," in *Proc. IEEE Military Communications Conference (MILCOM),* 2004.

[71] Scalable Network Technologies, "QualNet," Available HTTP: http://www.scalable-networks.com, Feb. 2006.

[72] S.J. Lee, E.M. Belding-Royer, and C.E. Perkins, "Scalability study of the ad hoc on-demand distance vector routing protocol," *International Journal of Network Management,* vol. 13, pp. 97-114, 2003.

[73] X. Zhang and G.F. Riley, "Performance of routing protocols in very large–scale mobile wireless ad hoc networks," in *Proc. 13th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS),* 2005.

[74] X. Zhang and G.F. Riley, "Scalability of ad hoc on-demand routing protocol in very large-scale mobile wireless networks," to appear in *Simulation: Transactions of the Society for Modeling and Simulation International,* 2006.

[75] R. Bagrodia, R. Meyer, et al., "Parsec: a parallel simulation environment for complex systems," in *IEEE Computer,* vol. 31(10), pp. 77-85, 1998.

[76] R. Barr, Z.J. Haas, and R.V. Renesse, "Scalable wireless ad hoc network simulation," in *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks,* Ch. 19, pp. 297-311, CRC Press, 2005.

[77] R. Barr, Z.J. Haas, and R.V. Renesse, "JiST: an efficient approach to simulation using virtual machines," in *Software Practice & Experience,* vol. 35(6), pp. 539-576, 2005.